Developing a Generic Agent-Based Model to Explore Servicising Policy

by Kasper Kisjes

in partial fulfillment of the requirements for the degree of

Master of Science

in Systems Engineering, Policy Analysis and Management

at the Delft University of Technology,

to be defended publicly on Tuesday January 28th, 2014 at 3:00 PM.

Student number: 1360639

Committee Chair:	Prof. dr. ir. Paulien Herder
First Supervisor:	Dr. ir. Igor Nikolic
Second Supervisor:	Dr. Martijn E. Warnier
Third Supervisor:	Dr. Reinier A. C. van der Veen

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Contents

Summary

Continuous economic growth, ignoring the incidental recession, is currently still coupled with increases in the use of resources and generation of wastes. The European Commission (EC) is looking for ways to achieve absolute decoupling between economic growth and environmental impacts. A shift from product-based to function-based production and consumption, known as 'servicising' of the economy, has the potential to contribute to absolute decoupling. The EC is therefore looking for policy measures on all political levels that may stimulate a servicising shift and thereby contribute to absolute decoupling.

In this thesis, I propose a generic agent-based model to inform policy towards absolute decoupling, with a focus on the role of servicising. The model captures interactions between selling and buying 'agents' and can be parametrised for many different specific markets. It integrates rational and non-rational considerations, decision making on multiple levels of both producers and consumers, and resulting material flows and impacts, all in a generic way. Also, the model features sophisticated market research as a novel basis for the decision making of agents in an artificial market.

The model was developed following the methodology for developing an agent-based model proposed by Van Dam, Nikolic and Lukszo (Nikolic et al., 2012). A substantial part of this thesis is reserved for a reflection on the methodology. The main conclusion from that part is that although the methodology provides valuable structure to help new modellers through model development, the recommended techniques and practices are mostly suitable for relatively small, domain-specific models. Additional practices are recommended in order to successfully build large and generic models.

The proposed model is suitable for the three planned case studies in the pan-European SPREE project, of which the generic model development constituted a central part. The concluding sections of this thesis provide suggestions for future extensions of the model, including the inclusion of social networks, spatial explicitness and chain-level interactions.

Acknowledgements

When looking for a suitable topic for this project somewhere at the end of 2012, I already knew that I wanted to do something with agent-based modelling, preferably with a focus on socio-economic aspects. Given his role as 'Mr. ABM' at the faculty (with ABM standing for agent-based-modelling), and after previous collaboration during my bachelor thesis project, Igor Nikolic was clearly the person to sit down with. Luckily, he was just looking for a researcher for the SPREE project, a large pan-European project that did not only have strong interdisciplinary and socio-economic aspects but also revolved around the development of an ambitious agent-based model. Although the role as ABM-modeller was actually meant for a post-doc, I was still allowed to take on that role in the context of my master thesis project. The report in front of you is the result of that project, and I want to thank Igor Nikolic for his confidence in me and for offering this exciting opportunity. Also, I want to thank Igor for always having his door open to discuss modelling and project matters while enjoying a cup of some fancy exotic tea.

Most of the modelling work I carried out in collaboration with Reinier van der Veen, whom I want to thank for his sharp mind during the modelling process and for proofreading every chapter of this thesis with equal attention and thoroughness. I would also like to thank Paulien Herder and Martijn Warnier for providing highly valuable feedback on the first version of this thesis. In addition, I want to thank all SPREE project team members that provided input for the model and thereby also made the end result of my master thesis project that much more exciting.

Finally, I would like to thank my parents, especially my father, for taking the time to read through my entire thesis multiple times and suggest numerous improvements.

Kasper Kisjes

Delft, January 14, 2014

1 Introduction

1.1 Unsustainable growth

This thesis concerns, above all, sustainable economic development. Continuous worldwide economic growth is still correlated with increasing use of consumed resources and associated wastes (Eurostat, 2011). On average, each person consumes nearly 15 tonnes of material resources per year (Lutter et al., 2009). Consumed materials either end up as 'stuff' accumulated in the economy or are converted into emissions or waste. The associated production and consumption processes cause increasing amounts of damage to ecosystems and health, not just in Europe but worldwide (European Environment Agency, 2012).

Figure 1.1 shows that although resource efficiency increases (shown as *decreasing* material intensity), resource extraction in absolute terms is still increasing. The figure shows a *relative* decoupling between economic growth and resource use, but not an *absolute* decoupling.

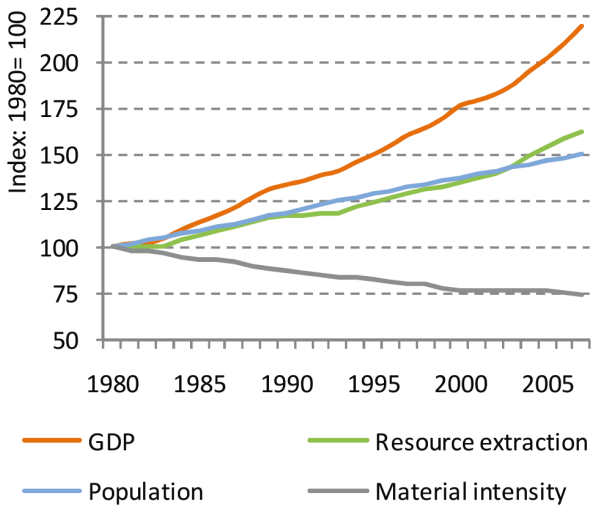

Figure 1.1: Worldwide economic growth and resource use (1980 to 2007. Source: Lutter et al. (2009))

One of the primary long-term policy goals of the European Commission is to make the European economy more sustainable, decoupling economic growth and welfare from the increasing use of resources (European Commission, 2012). In the EC's pursuit of this target, the Commission hopes to make Europe more competitive and inclusive, providing a high standard of living while causing much lower environmental impacts. These ambitious goals can only be achieved with considerable changes in production and consumption patterns.

1.2 Servicising

1.2.1 A shift from products to functions

One concept that may contribute to achieving absolute decoupling is 'servicising' of the economy. Servicising means that business models shift from *selling products* to *providing the functionality* associated with these products as a service (Rothenberg, 2007). Examples of servicising transactions include car sharing instead of car ownership, integrated pest management instead of each farmer buying his/her own machinery, and recycling-focused carpeting services instead of single-use carpets that are disposed of after their first period of use.

These examples have in common that they provide a more resource-efficient alternative to traditional ownership-based consumption models. They enable increased quality through sharing and create new incentives for manufacturers to improve the durability and re-usability of products. The examples illustrate that servicising has the potential to change the economic, social and environmental effects of production and consumption patterns:

- Economically, a more efficient use of resources must eventually translate to economic efficiency.
- Socially, servicising removes entry barriers to certain functionality. Also, shared costs of product use supports higher quality products for a lower price. These two effects can increase general quality of life.
- Environmentally, reductions of resource extraction and waste production reduce the footprint of production and consumption on the planet Earth.

1.2.2 Servicising policy

Because of these potential benefits, the European Commission is interested in exploring more elaborately if, and how, servicising of the economy can contribute to absolute decoupling. If the potential can be

confirmed, the European Commission is looking to formulate policies to stimulate servicising shifts in such a way that they provide economic, social and environmental benefits and contribute most effectively to absolute decoupling. Ideally, the Commission could use supportive tools to gain quantitative insights into the effects of possible policies.

1.3 Agent-based modelling and simulation (ABMS)

1.3.1 Complexity around servicising

Servicising represents a complex economic phenomenon. The adoption of servicising, as well as any societal effects, depends on economic *behaviour* of producers and consumers. These organisations and individuals are all interrelated in complex relationships that develop dynamically over time. Businesses have unique strategic beliefs, while consumers have varying lifestyles involving heterogeneous habits, norms and values. Consumption choices are rarely rational, at least in the traditional sense of cost minimisation.

1.3.2 AMBS: A suitable tool

Providing quantitative insights into the effects of policy related to such a complex phenomenon as servicising is no easy task. As argued in more detail later in this thesis, agent-based modelling and simulation (ABMS) is one of the only suitable tools to capture heterogeneity, relationships between individual actors and non-rational preferences and behaviour all at once. The method quickly gains popularity as a tool to support scientific research (see figure 1.2).

A common definition of an 'agent' in the context of ABMS is the following:

> "An Agent is a persistent thing which has some state we find worth representing, and which interacts with other agents, mutually modifying each other's states. The components of an agent-based model are a collection of agents and their states, the rules governing the interactions of the agents and the environment within which they live."
> (Shalizi, 2006)

By simulating the interaction between agents, ABMS allows to investigate production and consumption patterns on a system level based on assumptions on the heterogeneous properties, motivations and behaviour of individual businesses and consumers.

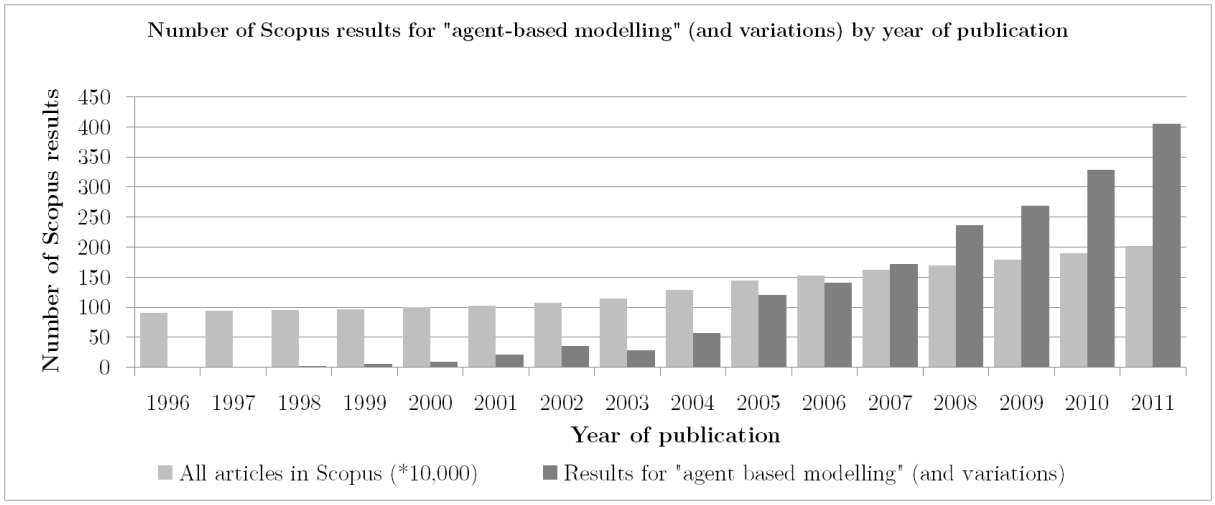

Figure 1.2: Number of articles on ABMS published per year according to search database SCOPUS

1.4 The SPREE project

1.4.1 Central aim

It is because of the potential of servicising, and the possible role of ABMS in informing servicising policy, that the European Commission decided to fund the international project SPREE (Servicising Policy for Resource Efficient Economy). Within SPREE, a diverse set of 10 institutions from 7 EU and associated countries collaborates for a period of three years to combine and extend knowledge on sustainable economy, policy design, agent-based modelling and domain knowledge in multiple sectors of the economy. The central aim of the project is to identify effective packages of potential servicising policies. Appendix A contains more information on the SPREE project.

A novel aspect of SPREE is the use of ABMS to compare policy effects on the absolute decoupling of economic growth and resource use, while achieving social benefits (SPREE, 2012). The SPREE ABMS model will represent an artificial market of sellers and buyers where servicising shifts can occur. The model is intended to measure economic, social and environmental effects of servicising-related policies that somehow influence the participants or conditions on the artificial market.

1.4.2 The SPREE *generic* model

Many studies that involve modelling and simulation in order to improve decision making represent a specific problem in a specific system. Examples include simulation of turn-around times at Schiphol airport, fresh-

water shortages in the Dutch water system and price volatility in the Dutch energy market. When modelling such problems, it makes sense to structure the model around the specific properties of that system. In the examples, such properties could be airport layout and facilities, river courses and salt intrusion fronts, or energy regulations and the merit order system of power plants. This makes the model suitable for the problem, but for that problem only: adapting it for application in another sector or country is difficult because of all the (sometimes hidden) structure and assumptions that do not apply in another sector or domain.

Contrary to such simulation models, SPREE aims to develop a *generic* model that provides a 'template' for simulation runs. This template can then be used for a wide array of policies, applied to B2B and B2C markets in virtually any sector and in any country, as long as the numerical data are available. Besides this, it is the ambition within the SPREE project to include many relevant concepts to servicising, such as innovation, heterogeneity between actors, bounded rationality, investment decisions, market research and pricing, resources efficiency, contracts, infrastructure availability, loyalty and skills. Developing a model that incorporates all this, while being generally applicable throughout sectors and countries, presents a major challenge. It forces the modeller to design a complex yet robust structure that integrates knowledge from experts from many research fields and can cope with all (or at least many) different circumstances and exceptions of the case studies it might be used for. The model development therefore constitutes one of the biggest challenges in SPREE, and was the central challenge during my thesis project.

1.5 Thesis project definition

1.5.1 Activities in the SPREE project

The faculty Technology, Policy and Management of Delft University of Technology has extensive experience of ABMS in various domains. A small TU Delft team, including me, was therefore selected to develop the agent-based simulation model within SPREE. Figure 1.3 illustrates the main activities of the TU Delft team in relation to activities carried out by other participants in the SPREE project.

The left-hand side of the figure shows the activities with explicit contributions from the TU Delft team, all relating to the model development process. This process consists of multiple activities, each one interrelating with other activities in SPREE. The first activity that relates specifically to the model development concerns model conceptualisation. Although primarily the responsibility of the TU Delft team, this activity depends on close collaboration with other partners within SPREE. During model conceptualisation, the TU Delft team builds on the information that other partners collect, and on the relevant economic, social and environmental measures that they identify. Collection of background data is therefore largely a task for other SPREE

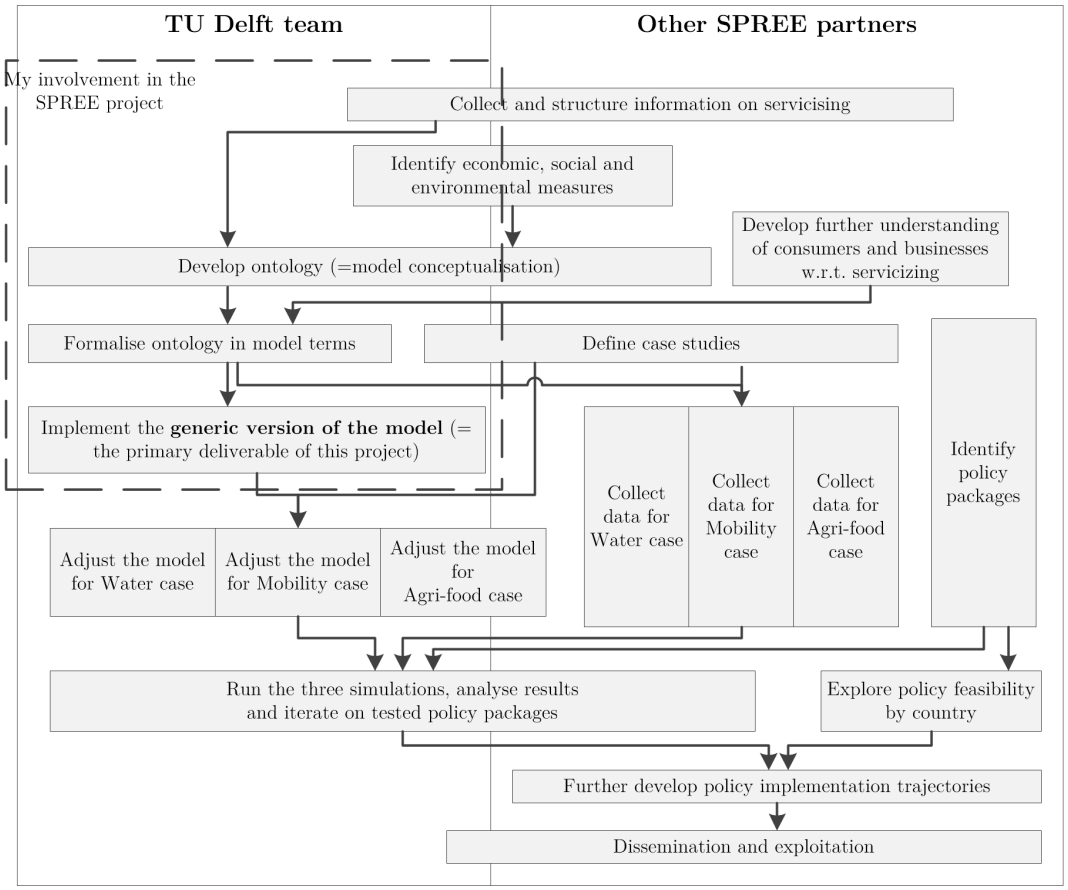

Figure 1.3: TU Delft work within the SPREE project

partners, whereas the TU Delft team is responsible for the translation into anything model-like.

In parallel to the model conceptualisation, SPREE experts on the behaviour of consumers and businesses develop a further understanding of behaviour of market participants in relation to servicising. Their findings are taken into account by the TU Delft team when translating the ontology (resulting from the model conceptualisation) into a more formal model definition.

The TU Delft team then implements the model formalisation in a software environment, resulting in the 'SPREE generic agent-based simulation model'. After implementation of the generic version of the model, the TU Delft team develops three sector-specific simulation models based on the case study definitions. In parallel, other SPREE teams start collecting case study data and designing packages of possible policy measures.

The sector-specific simulation models, once populated with collected data, can then be used to explore effects of identified policy packages and provide input for further development of the packages.

1.5.2 Research question

For the duration of my master thesis project, I was part of the SPREE team at TU Delft. As figure 1.3 indicates with a dashed demarcation line, I was involved in the development of the SPREE generic simulation model. Specifically, being the only one under the project leader with prior ABM experience, I had a substantial role in the conceptualisation phase and took the lead during the formalisation and implementation phases of the generic agent based model. In this role, I participated in international workshops and maintained frequent contact with other SPREE teams. These activities ensured optimal use of the input of other SPREE experts during the model development, and advised the project team on progress at each stage.

Given my role within SPREE, this thesis has a clear deliverable in the form of a working generic model. The central research question can be formulated as follows:

What should a generic Agent-Based Model look like that can measure social, economic and environmental impacts of servicising policies?

The model to be developed in this thesis and in the context of the SPREE project will be referred to as either the 'SPREE generic model' or simply the 'SPREE model'. The SPREE generic model focuses on the choices of producers and consumers, on the impacts of these choices and on the potential of policies to affect these choices in a favourable way. The model does not capture policy implementation trajectories.

1.5.3 A novel approach

To the best of my knowledge, few market-oriented agent based models have yet been reported in the literature that unite such a variety of concepts or are as generically applicable as is the ambition for SPREE generic model. For sure, none have been backed by comparable amounts of data on consumer and business behaviour in several sectors of the economy. The relevance of developing this model is therefore not limited to the practical/societal aspects associated with informing EU policy towards a more sustainable economy. The project also contributes to the scientific field of ABMS, stretching the boundaries of what can be modelled within this discipline.

1.6 Approach

1.6.1 ABMS methodology

Despite the increasing popularity of ABMS in the scientific community (see figure 1.2), there is not yet a generally accepted and applied methodology to guide the full development cycle of an agent based model. A recent book edited by Van Dam, Nikolic and Lukszo proposes such a methodology (Nikolic et al., 2012). Nikolic et al. provide "practical guidelines, examples drawn from published models and recommendations for specific tools that have proved to be useful in collaborative model development".

1.6.2 Steps of the chosen methodology

The main idea behind the methodology of Nikolic et al. is to force the modeller to not directly jump into model code, but instead follow a structured path through ten separate activities. Figure 1.4 depicts these activities as the ten steps of the methodology.

✓ Step 1:	Problem formulation and actor identification
✓ Step 2:	System identification and decomposition
✓ Step 3:	Concept formalisation
✓ Step 4:	Model formalisation
✓ Step 5:	Software implementation
✓ Step 6:	Model verification
Step 7:	Experimentation
Step 8:	Data analysis
Step 9:	Model validation
Step 10:	Model use

Figure 1.4: The 10 steps of the methodology proposed by Nikolic et al. (2012).

The first six steps are most relevant throughout this thesis (descriptions based on Nikolic et al. (2012)):

1. *Problem formulation and actor identification.* The need for a model typically derives from a certain problem. The first step in model development is to understand exactly what the problem is, including the exact lack of insight, how the observed pattern deviates from the desired pattern, and some hypothesis on the main causes for that deviation. Also, it is important to understand whose problem it is, which other actors are involved and what the role of the modeller is.

2. *System identification and decomposition.* With the problem identified, the next step in the methodology is to make an inventory of all concepts, actors and objects, behaviours, interactions, flows and properties that may be relevant to the problem. The modeller should then structure all the identified elements and the relations between them. Initial structuring is followed by an iterative process of simplification and elimination where possible, such that only the essential elements remain. The outcome of step 2 is a clear, structured set of elements. In the case of the SPREE model, it provides answers to questions such as:

   - Which agents (e.g. producers, consumers) and objects (e.g. products, services) are essential?
   - What decisions (e.g. change price, switch to a different consumption model) do the agents make and on what basis (e.g. price vs 'quality', minimum requirements)?
   - How do they interact (buying/selling, peer influence)?
   - What are the material flows in the model?
   - How should policies and external developments be taken into account?
   - How can we actually measure economic, social and environmental effects?

3. *Concept formalisation.* Although the modeller already makes important modelling choices in step 2, this does not yet extend to all the *relations* between elements. Also, the elements are still defined in a natural language that may be context-dependent and ambiguous. Such ambiguity is undesired in general and specifically because computers cannot deal with it. Step 3 therefore serves to relate the elements together and make the elements and relations from the previous step 'computer-understandable' in addition to human-understandable. The methodology proposes two options but encourages the modeller to create a formal 'ontology' that is basically a structured dataset of all model components and the relations between them. The ontology can be a very suitable basis for object-oriented software implementation.

4. *Model formalisation.* With the 'who' and 'what' specified in a formal ontology, step 4 requires the modeller to specify who does what and when. The first subtask here is writing a model narrative that describes (in a natural language) the process that agents 'experience' in the model. For example, one phrase in the narrative could be "The agent wakes up, gets a cup of coffee and checks if she still has an option to satisfy her need for mobility this day." The second and last subtask of step 4 is to translate the narrative into a pseudo-code, which is essentially an algorithm written in human-readable form. It forces the modeller to think about a logical model layout without being distracted by software

implementation details. Pseudo-code consists of computations, value assignments, iterations, loops, conditions and input/output operations, among possible other things.

5. *Software implementation.* Several good modelling environments are available for building agent-based models, including NetLogo (open source), RePast (open source) and various commercial platforms. Good practices during the software implementation step include version control, elaborate code documentation, naming conventions, division of tasks and responsibilities, and bug tracking.

6. *Model verification.* Before moving on to model use, the modeller should verify that he or she 'built it right', i.e. correctly translated the conceptual model into model code. The methodology proposes several methods for model verification, such as the recording and tracking of agent behaviour, single-agent testing, interaction testing and multi-agent testing. The verification process typically consists of testing hypotheses on agent and model behaviour, given certain inputs or circumstances. The more tests the modeller carries out, the more certain he/she can be that the model was built right.

Given the position within the SPREE project, as a prelude to the three case-specific studies, I will not perform any experiments with real-world data and will not be able to draw any conclusions about the effectiveness of specific policies. These activities, related to steps 7-10 of the methodology, will happen later in the SPREE project.

The methodology helps to avoid extensive revisions at the point where they would hurt most: during model implementation. The clear stepwise approach does not mean that the modeller can never revisit previous steps: the development of a complex agent based model is inherently an iterative exercise.

1.6.3 Role of the methodology in this thesis

The methodology proposed by Nikolic et al. appears to be a promising tool for ABMS development that is unique in its practice-oriented approach. I will therefore use this methodology (from here on simply referred to as 'the methodology') throughout the complete development cycle of the generic model up to model implementation and partial verification and validation (i.e. step 1-6 of the methodology). Although the methodology captures best practices of previously developed models, there may still be room for improvement. Mainly as an additional contribution to the field of ABMS, the final chapter of this thesis therefore includes an elaborate reflection on my experience of using the methodology for the development of the SPREE generic model.

1.7 Thesis Structure

1.7.1 High-level overview

This thesis serves as a guide through the development process of the model. It provides context and explains the choices made throughout the process. The thesis also presents the resulting model and explores model behaviour insofar possible without case study data. Figure 1.5 on the next page illustrates the subdivision of this thesis into chapters, as well as the storyline followed through each of the chapters.

Looking at a high level, chapter 2 provides context by introducing servicising in some more depth, after which chapter 3 discusses why ABMS is most suitable for informing policies on servicising. Chapters 4-8 concern the actual model development. They describe the developed SPREE generic model and thereby together provide an answer to the central research question. Chapter 9 summarises this answer by highlighting the main features of the model and discussing possible extensions. Finally, Chapter 10 provides an elaborate reflection on the chosen methodology, on the provided functionality of the resulting model and on the process in general.

1.7.2 Relation with the methodology

Chapters 4-8 correspond (as a whole) to steps 1-6 of the methodology. Looking at the relation between the chapters and the methodology, chapter 4 addresses step 1 in the methodology (figure 1.4) by providing a revised, more extensive problem formulation. It also derives a set of key concepts and desired model functionality based on the findings from chapters 1-3. This latter part of the chapter (i.e., key concepts and functionality) relates to the first part of step 2 of the methodology (i.e., system identification). Chapter 5 explores how other published ABMS studies have addressed such concepts, and discusses how suitable their approaches may be for the SPREE model. Chapter 6 presents the actual model description of agents, states and interactions, all supported by illustrative diagrams. It thereby addresses the second part of step 2 (i.e., structuring), as well as steps 3 and 4 of the methodology. Chapter 7 presents the finished model after implementation (step 5) and discusses performed verification tests that build confidence that the model was built right (step 6). Chapter 8 presents some possible model outcomes based on fictitious data from a fictitious case. This illustrates the type of results that can be expected from steps 7-10 of the methodology, when the model will be populated with actual data during the three case studies in the SPREE project.

	Thesis outline	Storyline
Background and methodology	1. Introduction	- Background: EC goals, servicizing, SPREE introduction, ABM introduction - Project definition: research question, methodology, scope and limitations - Thesis structure
	2. Servicizing	- What is servicizing? - Why servicize the economy? - What are the main obstacles to servicizing?
	3. The choice for ABM	- Why do we need a simulation model? - Why should it be ABM? - Which are possible downsides that we need to be aware of?
Model development	4. Key concepts and functionality	- What is the exact lack of insight that we want to address? - Which are key concepts to servicising that the model should somehow include? - What functionality must the generic model provide?
	5. Related ABMS works	- What related work exists on agent-based modeling of artificial markets? - How are identified concepts addressed by other scholars? - Which are promising unexplored dIrections in the field of ABMS artificial markets?
	6. High-level modelling choices	- On a high level, how will the model address the key concepts and functionality? - What elements does the model consist of, and how do they interact in model procedures? - Which are inherent limitations of the chosen approach?
	7. NetLogo implementation	- What does the implemented model look like? - How do we know that we built it right? - Which refinements have yet to be implemented?
Model behaviour	8. Model behaviour	- Can we explain what's happening, even if the behaviour is counterintuitive? - Does the model meet all desired functionality? - Is the model suitable for use within (and outside of) the SPREE project?
Reflection/ conclusion	9. Conclusion	- What are the main features of the designed model? (answer to the 1st research question) - How suitable is the Van Dam methodology for ABMS development? (answer to 2nd RQ) - What else have we learned?
	10. Reflection	- How suitable was the followed methodology, and how can it be improved? - Did I make the right choices? - What is the main contribution of this thesis?

Figure 1.5: The outline and storyline followed in this thesis

2 Servicising

2.1 Introduction

The SPREE project (as well as this thesis project) revolves around 'servicising', a relatively new term for a type of economic interaction that can help achieve economic, social and environmental goals. Despite the theoretical benefits and despite a number of success stories, servicising has not yet been adopted on a large scale. The reason may be simply that servicising is not as viable in reality as its proponents believe it to be. Alternatively, the lack of actual servicising success stories may indicate a market failure. In this case, servicising may really have large potential, but current market conditions prevent it from gaining critical mass. Government interventions may then be justified to help bring the market towards a dominant equilibrium.

The central research question of this thesis concerns the development of a simulation model to explore effects of such interventions. Providing essential background for the model development, this chapter introduces servicising in some more depth (section 2.2) and discusses examples and studies that illustrate its potential (2.3). After section 2.4 presents suspected causes for the slow adoption rate, section 2.5 discusses the type of policies that may stimulate servicising and thereby contribute to a more sustainable economy. The findings in this chapter are the basis for the choice for ABM in the next chapter and provide input for modelling choices made in chapters 4-8.

2.2 Definition of servicising

2.2.1 The definition itself

The notion of servicising was first introduced by White et al. in 1999, who describe this concept as "the emergence of product-based services, which blur the distinction between manufacturing and traditional service sector activities" (White et al., 1999). The SPREE project uses a refined definition as formulated by

Oksana Mont, who has published several fundamental papers on servicising and is also one of the participants in the SPREE project. In Mont's definition, a servicising system represents 'a system of products, services, networks of actors and supporting infrastructure that continuously strives to be competitive, satisfy customer needs and have a lower environmental impact than traditional business models' (Mont, 2004). In Mont's definition, servicising is more than just products being replaced by services. It must also be competitive and provide social and environmental improvements. For example, Mont's definition would not classify software leasing instead of selling as 'servicising': it does not deliver any relevant environmental benefits.

2.2.2 Some examples

According to Mont, servicising typically involves 'functional thinking'. In order to reduce material throughput in the economy, emphasis should be on *functions provided*, not on material products sold (Mont and Lindhqvist, 2002). Clear examples of functional thinking in relation to servicising can be found in the product use phase: sharing (e.g. car sharing), leasing (e.g. carpets), pooling (e.g. farming equipment) or renting (e.g. bike renting). But servicising also extends to areas such as support services, sale services, maintenance services, end-of-life (EOL) services and collaborative consumption (Mont, 2004). Baines et al. (2007) provide some successful real-world examples of servicising, which are listed in table 2.1.

Organization	Description
Xerox International	Products are sold guaranteeing fixed price per copy from products/processes designed for remanufacturing.
Parkersell (UK)	Parkersell developed a product service integrated lighting system solution for Sainsbury's more efficient in life cycle costing and environmental improvement.
Castrol Inc. (USA)	Lubricant service packages reducing lubricant consumption. Profit from cost saving not consumption.
Eastern Energy (UK)	Not just energy. Energy management, consumption and process monitoring and utility awareness and training.
Electrolux (Sweden)	Initial fee then pay per wash from remotely monitored energy efficient machine and launderette system solutions including maintenance, repair and finance services.
Mobility (Switzerland)	Vehicle sharing group - 1400 cars, 850 locations, 350 communities.

Table 2.1: Successful examples of Product-Service Systems (adapted from Baines et al. (2007))

2.2.3 A structural change

The presented definitions and examples illustrate that servicising changes consumption and production patterns. Characteristics of servicised business models are a distinction between user and owner, a shift in incentives for behaviour and choices during design, production and consumption processes, and/or shifted responsibility for maintenance, operation and end-of-life processes. A proper understanding of these aspects is vital for policymakers wishing to use servicising for societal goals (European Environment Agency, 2010). Mont's definition of servicising suggests that such understanding requires the bundling of knowledge from experts in a wide variety of disciplines, as happens in the SPREE project.

2.3 Promises of servicising

2.3.1 Reduced resource use

The main reason for the EU to look into servicising is its potential to contribute to a decoupling of economic growth (or wealth creation) from consumption of natural resources (Mont, 2000). This can be achieved by shifting from *ownership-based* to *functional* economic thinking. All the examples in table 2.1 show similar or increased functionality requiring fewer resources per unit delivered, such as number of printed copies or km of transportation. When consumers and producers start thinking in term of functions provided instead of ownership and adopt servicising on a large scale, there are multiple effects that help reduce resource use:

- Products are used more intensively, decreasing the resource impact per functional unit delivered.
- Producers have new economic incentives to design durable goods instead of products with built-in obsolescence.
- Producers also have an increased incentive to optimise maintenance and improve end-of-life (EOL) processes.

2.3.2 Other benefits

Another benefit of servicising is that it can make certain functions more accessible. Ownership-based consumption requires a significant investment of money by the consumer, even if the product is intended for low-frequency or even one-time use. Servicised products (or rather *value propositions*) can be paid for per unit of functionality delivered and are suitable for frequent as well as low-frequency use. This increased access to functionality potentially increases consumers' quality of life. In certain domains, servicising may

also contribute to the quality and general efficiency of the delivered functionality, as the offering business may have skills that the consumer (or consuming business) does not possess. As an example, integrated pest management offered by specialised businesses is likely to achieve better results than farmers operating their own pest management equipment (Flint, 2012). As a final possible benefit, some forms of servicising have proven to improve social cohesion in local communities. (Devisscher and Mont, 2008).

2.3.3 Promising domains

Servicising is especially promising in the areas of housing, transport and food, which together are responsible for 70% of the environmental impact of consumption (Tukker and Jansen, 2006). The European Environment Agency also recently emphasized the importance of the consumption of goods and services as a key driver of global resource use and associated environmental impact (European Environment Agency, 2010). Various case studies put the average reduction of environmental impacts of servicised systems at 50-70%. These case studies include shared washing centres (Weaver et al., 2000), car sharing (Sperling et al., 2000), ski rental services (Hirschl et al., 2003) and integrated pest management (Schmidt-Bleek and Lehner, 1998).

2.4 Main obstacles to servicising

Given all of the advantages listed above, one might well expect the world to have already adopted servicising on a large scale. This is not the case: the examples in table 2.1 are merely exceptions to the 'rule' of ownership-based consumption. Experts from the SPREE project have identified a number of current market conditions that present an obstacle to large-scale adoption of servicising. These unfavourable conditions on both the consumer and business sides represent important concepts within the SPREE project and within this thesis.

2.4.1 Consumer-related obstacles

Many scholars observe that consumers are not as 'rational' (i.e. purely driven by economic considerations) as most economic theory assumes them to be (Schwartz et al., 2002; Choo and Mokhtarian, 2004; Briceno and Sagel, 2006). Consumption serves not only to fulfill functional needs, but also to achieve social objectives (Silver, 2002). Status, peer pressure and lifestyle consistency are examples of such social objectives that strongly influence consumption choices (Evans and Jackson, 2007). Other not-so-rational considerations include pleasure in product use, familiarity with the product and brand loyalty (Jordan, 1998). Most of these aspects (except for environmental awareness) typically favour ownership-based consumption choices

and may for a large part explain why the general public adopts servicising only slowly, if at all. Aspects that servicising scores well on, such as the carbon/resource/toxic profile of a product, hardly play a role in consumers' decision processes, except when it is part of the lifestyle they choose (Kotakorpi et al., 2008).

2.4.2 Business-related obstacles

On the business side, entrepreneurs appear hesitant to switch to servicising-based business models. Low consumer acceptance limits sales. Without a moderate to dramatic paradigm shift, many servicising-based business models are just not viable (Fishbein et al., 2000; Toffel, 2002). New business models, perhaps based on innovative product-service systems, may be needed to make servicising successful. A related problem is of a financial/financing nature: servicising in some sectors requires vast investments in infrastructure and/or ICT investments that cannot be justified by the current level of expected reward (White et al., 1999). If nobody takes the first step, servicising may never become successful in such markets.

2.5 Servicising policies

2.5.1 Justification of policy instruments

Given the potential benefits of a shift towards servicising and the inability of the market to achieve this shift all by itself, it seems justified to develop and use policy instruments that stimulate a shift towards servicising and thereby towards absolute decoupling of resource use and economic growth. The European Commission has asked the SPREE project to identify such instruments and explore their possible economic, social and environmental effects. Possible servicising policies include monetary incentives to producers, selective permits for business activities, and informational campaigns (European Environment Agency, 2010).

2.5.2 Unintended effects

Complementary to the identification of policies, policy makers must be aware of unintended policy effects. Toffel identifies risks related to the transaction costs of servicising transactions, such as bilateral dependencies and bilateral monopolies coming into existence, adverse selection issues and even moral hazard (which in this case relates to the distinction between users and owners) (Toffel, 2002). Servicising is not some risk-free instant fix - structural changes in economic patterns are a sensitive matter with a potentially large societal impact. In order to prevent unintended negative effects of policy instruments and achieve possible synergy, policy should be thought of in terms of 'policy packages': "combinations of individual policy measures, aimed at addressing one or more policy objectives, created in order to improve the effectiveness of the

individual policy measures, minimise possible unintended effects, and/or facilitate interventions' legitimacy and feasibility." (Givoni et al., 2011).

2.6 Conclusion

Servicising encompasses a shift in economic behaviour, from an ownership-based to a functional orientation of both consumers and producers. Servicising can contribute to absolute decoupling of resource use and economic growth, as well as other societal objectives. Despite its potential, there are still few examples of successful servicising-based business models. Possible causes include non-rationality in consumer behaviour and a scarcity of viable business models. The potentially beneficial effects may justify targeted policies. Well-designed packages of policy instruments can maximise policy effectiveness and minimise adverse side effects. The next chapter discusses why agent-based modelling and simulation offers a very suitable tool to help in the design and evaluation of such policy packages.

3 The choice for ABM

3.1 Introduction

The previous chapter has provided more insight in servicising and the potential of servicising to contribute to absolute decoupling and economic growth. It concluded with a call for servicising-oriented *policy packages* that may help achieve social, environmental and economic goals through servicising. After the first task of identifying a list of possible policies comes the very difficult task of screening those policies for effectiveness and implementability. The results of this evaluation eventually provide the input for the design of effective policy packages. The present chapter makes the argument that simulation, agent-based modelling and simulation in particular, provides an extremely helpful tool to assess the effectiveness of possible policy instruments and policy packages (sections 3.2 and 3.3). Section 3.4 addresses some known downsides of this modelling approach and evaluates the implications for the SPREE generic ABM.

3.2 Why do we need a simulation model?

3.2.1 A complex system

Servicising policy aims to change the economic *behaviour* of consumers and producers on a large scale. As chapter 2 has illustrated the proposed study of economic behaviour and policy effects involves a significant amount of complexity. Economic behaviour is influenced by a wide range of individual preferences of market participants. Many of these preferences are not based on (economically) rational considerations. This is especially tru for consumers, who may have various preferences that have nothing to do with economic optimisation. But even the choices of producers depend partly on unique company identity, culture and beliefs (He and Balmer, 2013). Such non-rational factors cannot be easily quantified and put into simple calculation models. Adding to the complexity, neither consumers nor producers have full market knowledge, and different (groups of) consumers or producers have distinctly different properties and behaviour. This

heterogeneity can strongly affect policy effects.

3.2.2 Unpredictable effects

The presence of substantial complexity means that it is virtually impossible to correctly predict how a large heterogeneous population of market participants will respond to novel policy instruments. It also implies that any external change, such as the effectuation of policy instruments, may have unforeseen side effects that may even counteract the intended policy effects. Policy makers, consulting experts and academic scholars have already gained advanced insights in the complexity involved. These include insights in aspects such as the main consumer and producer subgroups in certain markets, their main beliefs and the interdependencies between them. However, as there are many interrelations between all complexities involved, these individual insights alone will not suffice to fully understand policy impacts in a complex system such as a real-world market of competing products and services. They need to be combined into one integral model, that preferably allows for dynamic exploration over time.

3.2.3 Computer simulation

Computer simulation offers a powerful tool to explore the behaviour of complex systems consisting of elements that influence each other in many direct and indirect ways. It forces decision makers and researchers to provide concrete and precise definitions of their understanding of the elements of a system, the behaviour of these elements and the interactions between them. This process of collaboratively developing a system conceptualisation that represents a shared understanding of the system is a very insightful exercise all by itself. It also allows to actually *simulate* over time some complex interactions in the system, in a varying context of policy and external developments. Depending on the quality of defined inputs, the outcomes of simulation-based analysis can provide unique insights in system behaviour, policy robustness and unintended side effects that would be unattainable otherwise. For example, such insights may include the observance of previously undetected feedback loops or the identification of parameter ranges where a policy is effective and where it is not.

3.3 Why should it be agent-based?

3.3.1 Three modelling paradigms

The three primary paradigms for the modelling and simulation of complex systems include System Dynamics (SD), Discrete Event Simulation (DES) and Agent-Based Modelling and Simulation (ABMS) (Maidstone,

	System Dynamics (SD)	Discrete-event Simulation (DES)	Agent-based Modelling and Simulation (ABMS)
Focus on:	System properties (top-down)	Process (top-down)	Individual agents and objects (bottom-up)
Micro-level entities:	None	Passive entities	(Inter)active entities
Driver of dynamic behaviour:	Feedback loops	Event occurrence	Agents' decisions and interactions
Mathematical formalisation:	Stocks and flows	Events, activities and processes	Agents and environment
Handling of time:	Continuous	Discrete time steps	Discrete time steps
System structure:	Fixed	Fixed	Dynamic

Table 3.1: Three central paradigms in computer simulation of complex systems (adapted from Behdani (2012))

2012). Table 3.1 compares some key characteristics of these distinctive approaches to computer simulation.

3.3.2 SPREE requirements for a modelling approach

So, which of these modelling paradigms best suits the needs of the SPREE project? The answer should follow from what the modeller considers the most important system properties that must be captured to understand the problem at hand. In this case, the most important system properties follow from the description of servicising and the identified obstacles to large scale adoption as discussed in chapter 2. The obstacles all involve consumer and producer *choices* and *behaviour*. The behaviour and the underlying considerations vary between, and even within, subgroups of market participants. This *heterogeneity* within the study population makes it hard to predict the effect of any policy instrument on the population as a whole. Consumers and producers organise themselves - not necessarily intentionally - in network *structures that dynamically change* over time.

3.3.3 ABMS as the best fit

When comparing the above description of important system features with the properties of the three simulation paradigms outlined in table 3.1, ABMS is clearly the best fit. SD copes badly with heterogeneity and assumes fixed system structures, whereas DES does not allow agents to make autonomous decisions or engage in interaction with other agents. ABMS allows the modeller to capture autonomous decisions made by a heterogeneous population of agents that interact in dynamically changing structures. For this reason,

ABMS is the simulation paradigm of choice for the SPREE project.

3.4 Downsides of ABMS

3.4.1 Substantial data requirements

The powerful properties of ABMS come at a steep price. Capturing the heterogeneous considerations and interactions of market participants is highly data-intensive (Tesfatsion, 2002). Every aspect included in the model needs underlying data distributions, possibly varying between and within agent subgroups. Given the large number of concepts that are relevant to servicising and servicising policy, it is easy to see how the data requirements may quickly explode. Fortunately, the SPREE project has the resources to gather substantial amounts of data. Still, some data pieces will unavoidably remain undiscovered. In such cases, the model will rely on assumptions that add uncertainty to the model outcomes. Thorough sensitivity analysis will be necessary to cope with this uncertainty.

3.4.2 Insufficient theoretical basis

Scientific insights in the behavioural processes of market participants have not yet developed to the point where one can make a convincing claim that an agent-based model is indeed realistic with respect to agent behaviour, even if it reproduces phenomena observed in the real world (Midgley et al., 2007). A large amount of real-world experimentation under highly varying conditions would be required to properly validate the way agents behave (Axtell, 2000). Although the SPREE project will attempt to find out as much as possible about consumer and producer behaviour, the model will inevitably depend on some strong assumptions here.

3.4.3 Validation challenges

As a third ABMS downside, the outcomes of ABMS experiments typically have the form of a distribution of outcomes, not necessarily resembling any theoretical statistical distribution. None of the broadly accepted current statistical practices support the validation of this type of outcomes against reality (Tesfatsion, 2006). This makes the validation of model outcomes much more difficult. On the other hand, one might see the distributed outcomes as a merit of ABMS. This type of outcome does not claim that there is just one possible future. It rather presents a range of possible futures that may follow from the input assumptions and parametrisation of the system.

3.5 Conclusion

The SPREE project concerns the exploration of servicising policies and their possible effects. The economic behaviour of heterogeneous groups of consumers and producers is a highly complex matter. Computer simulation offers a powerful tool to explore market interactions and policy effects under such complex conditions. Agent-based modelling and simulation (ABMS) is the only broadly accepted simulation approach able to cope with heterogeneity between agents and with dynamically changing network structures within the modelled system. Still, it is important to keep in mind that ABMS poses extreme data requirements, relies quite strongly on assumptions and generally proves difficult to validate. The next chapter will start with the actual ABMS model development.

4 Key concepts and functionality

4.1 Introduction

This is the pivotal chapter within this thesis, as it makes the step from problem background (chapters 1 and 2) and methodological choice for ABMS (chapter 3) to the actual development of the SPREE generic model. Section 4.2 addresses the first step of the methodology by reformulating very precisely the problem, the lack of insights and the behavioural patterns of interest that the model should address. Section 4.3 relates the problem identification and the background information on servicising (provided in chapter 2) to a number of key concepts that the model should capture. Finally, section 4.4 formulates desired model functionality: what mechanisms must the model include, what type of inputs must it be able to process and what questions must it help answer?

4.2 Extended problem formulation

4.2.1 The modelling questions

Nikolic et al. stress the importance of a well-formulated problem. They present the modeller with a series of questions to help create a deeper understanding of the problem:

- What is the problem?
  - What is the exact lack of insight that we are addressing?
  - What is the observed emergent pattern of interest to us?
  - Is there is a desired emergent pattern, and if so, how is it different from the observed emergent pattern?
  - What is the initial hypothesis on how the emergent patterns emerge and why the observed and desired emergent patterns differ?

- Whose problem are we addressing?
- Which other actors are involved?
- What is our role?

The remainder of this section presents the answers to these questions as formulated for the SPREE generic model. The text is a revised version of the outcomes of one of the initial SPREE workshops, where the questions were discussed. The questions and their answers constitute the first step of the actual model development process.

4.2.2 What is the problem?

The *problem* lies in the fact that the European economy has not yet succeeded in the absolute decoupling of economic growth (higher business and consumer value) from accompanying environmental impact (resource use, pollution, rebound effects). Servicising systems can potentially contribute to achieving absolute decoupling, but have not yet been established on a large scale and do not yet show a significant contribution to absolute decoupling. There is a *lack of insight* about whether servicising can contribute to absolute decoupling, the conditions under which this is the case, and the possible role for policies to help achieve such conditions. The *desired emergent pattern* is absolute decoupling, possibly as the result of large-scale adoption of servicising systems.

The *initial hypothesis* on the observed environmental degradation as a result of economic growth, and lack of absolute decoupling, is that our world undervalues environment. Positive environmental effects are usually just a by-product, subordinate to other considerations such as costs, status and pleasure. Market conditions somehow are not sufficiently favourable for large-scale adoption of servicising systems, let alone absolute decoupling, without policy intervention.

4.2.3 Whose problem are we addressing?

In general, by aiming for absolute decoupling we are addressing the sustainability problem of Mother Earth and "(global) society" in general. In the SPREE project, we particularly look at effects within the EU. The primary problem owner is the European Commission. However, since policy is often formulated on lower levels, national and even regional governments can also be seen as problem owners. This is especially true for the three case studies that SPREE includes, which all apply to a specific sector in a specific region or country.

4.2.4 Which other actors are involved?

The primary involved actors are businesses (more specifically: producers and sellers of products and services) and consumers. These actors constitute markets where a servicising shift might occur. Several layers of government (municipal, regional, national, international/European) are important as they can impose policy on the market to influence its dynamics and outcomes. Infrastructure providers may be important as facilitators of some forms of servicising. NGOs, the educational sector and the media can also play a role, especially in influencing consumers‘ attitudes.

4.2.5 What is our role?

Throughout the full ABMS track in SPREE, our (as the entire SPREE project team) role is to explore/design the servicising system components, gain new insights, analyse policy options and use the results to advise the problem owner on how to act adequately. When looking at my personal role within SPREE, the actual advise to the problem owner based on analysis with real-world data is out of scope. My specific role is first to extract, from the SPREE team and from the literature, the key concepts that influence the adoption of servicising systems and the effects of this adoption with regard to absolute decoupling. After the identification of important concepts, my role is to translate these concepts into a working agent-based simulation model of an artificial market. The model should produce realistic market behaviour, facilitate the exploration of various policies and be generic enough to be applied in basically any sector of the economy.

4.3 Key concepts

This section proposes a set of key concepts that the SPREE generic model should capture because they are so clearly relevant to servicising. The presented selection of concepts is the result of a continuous iterative process, although most concepts have been important from the start onwards.

4.3.1 Concepts relating to the servicising definition

Oksana Mont’s definition of servicising (see also section 2.2) will serve as the first source of inspiration: ‘(Servicising constitutes/involves) a system of products, services, networks of actors and supporting infrastructure that continuously strives to be competitive, satisfy customer needs and have a lower environmental impact than traditional business models’. At this point (given the previous chapters) it is clear that all of the underlined words represent important concepts to be included in the SPREE generic model. As a small extension,

a 'consumption model' can be seen as the consumer's counterpart of a business model. This already results in nine concepts to be included in the model:

Concepts based on Mont's definition of servicising:

Product	Service	Environmental impact
Business actor	Business model	Competition
Consumer actor	Consumption model	Consumer need

4.3.2 Concepts related to functional thinking

The notion of functional thinking, as opposed to an ownership-based approach, is also central in servicising literature, adding two more concepts to the list: ownership-oriented versus function-oriented production and consumption patterns.

Concepts based on the goal to inform policy:

Ownership-oriented	Function-oriented

4.3.3 Concepts related to servicising promises

The model should further capture some of the identified promises of servicising (as described in section 2.3). This means that it should explicitly address absolute decoupling of economic growth (or wealth creation) from the consumption of natural resources and the generation of associated waste. Economic aspects should include business profits and consumer spending. The model should also somehow address that servicising may improve quality of life. Finally, the effect of more specialised skill on delivered functional quality also represents an important concept for the SPREE generic model.

Concepts based on promises of servicising:

Business profits	Resource use	Quality of life
Consumer spending	Generated waste	Skill

4.3.4 Concepts related to servicising obstacles

The section on servicising obstacles (section 2.4) has introduced some concepts that may explain why servicising has not yet been adopted on a large scale. On the consumer side, social objectives and ownership-oriented lifestyle choices generally favour traditional consumption patterns. Brand loyalty and familiarity with a certain consumption model may restrain consumers from switching to a service-based provider of the sought-after functionality. On the business side, lacking consumer acceptance presents a major obstacle. Innovative business models may better position servicising in the market. Lacking infrastructure requires high investments that businesses often cannot afford on their own.

Concepts based on identified obstacles to servicising:

Social objectives	Lifestyle	Loyalty
Consumer acceptance	Innovation	Infrastructure

4.3.5 Concepts related to servicising policy

Arriving at the last category, the SPREE project has the eventual goal to inform policy. The model should therefore include policy instruments and their effects on the market. Moreover, the model should somehow address scenario uncertainty deriving from external market developments.

Concepts based on the goal to inform policy:

Policy instruments	Policy effects	External developments

4.4 Desired functionality

This section provides a specification of desired model functionality based on the problem formulation, the identification of relevant concepts, and discussions with SPREE partners.

4.4.1 General functionality

Starting with the basics, the SPREE generic model must:

- Represent an artificial market of producers/sellers and consumers.

- Simulate the behaviour of market participants over time.
- Allow for competition between products and services that can both satisfy some consumer need.
- Explicitly address the possibility of a shift towards servicising.
- Be generic enough for application in multiple domains, including at least mobility, agri-food and water use.
- Facilitate the exploration of policy effects:
  - Economic effects (business profit, consumer spending).
  - Social effects (quality of life).
  - Environmental effects (resource use, generated waste).

4.4.2 Functionality relating to business actors

Business actors produce and/or sell the products and services offered on the market. Desired functionality with regard to simulated business actors relates primarily to their decision-making considerations. Simulated business actors should:

- Have a unique set of beliefs/preferences about how they want to operate in the market (i.e., their 'strategy').
- Periodically re-evaluate both strategic (i.e., business model configuration, market positioning) and tactical choices (i.e., price/ volume setting), taking into account their unique beliefs.
- Consider structural changes in their business models (e.g. adopt a more servicising-based business model if the conditions are right).
- Make decisions in a 'smarter' way than just trial-and-error, taking into account important indicators such as market demand and (return on) investment.
- Not have full information on consumer's preferences, but at the same time not be completely unaware of them.

4.4.3 Functionality relating to consumer actors

Finally, consumer behaviour has so far been characterised as heterogeneous, diverse, non-rational and influenced by lifestyle choices. Simulated consumer actors should:

- Have a unique set (although possibly correlated within groups) of consumption needs, personal preferences and susceptibility to social influences (i.e., their 'lifestyle').
- Take into account choices made by other consumers when making their own choices.
- Take into account their own historic choices when making new choices, for example by favouring a previously selected supplier.
- Be able to filter on products and services based on certain thresholds (including social thresholds), but also be cognitively capable of making a good choice out of remaining possibilities based on their unique preferences.

4.5 Conclusion

This chapter has combined information from chapters 1, 2 and 3 to formulate an extensive problem formulation and to derive key concepts to be included in the SPREE generic model. Based on the problem formulation and the identified concepts, the chapter specified desired model functionality in terms of what the model should represent, what aspects of business actors it must address and what aspects of consumer actors. Figure 4.1 summarises the identified central concepts and desired functionality. The following chapter explores how other ABMS literature addresses these concepts and discusses what approaches are most suitable for the SPREE generic model.

Important model concepts		
General	Business-related	Consumer-related
Product	Business actor	Consumer actor
Service	Competition	Consumer need
Environmental impact	Business model	Consumption model
Resource use	Business profits	Consumer spending
Generated waste	Skill	Quality of life
Ownership-oriented	Innovation	Social objectives
Function-oriented	Infrastructure	Lifestyle
Policy instruments	Consumer acceptance	Loyalty
External developments		
Policy effects		
Desired functionality		
General	Business-related	Consumer-related
1. Artificial market of producers/ sellers and consumers	7. Heterogeneous sets of beliefs/ preferences of businesses	12. Heterogeneous lifestyle aspects of consumers (need, preferences)
2. Simulation of dynamic behaviour of market participants	8. Periodic strategic and tactical reconsiderations of businesses	13. Consumers take choices of other consumers into account
3. Competition between products and services	9. Allow for structural changes in business models	14. Consumers take their own previous choice(s) into account
4. Explicitly address shifts towards servicising	10. Businesses make smart decisions based on forecasts	15. Consumption choice based on thresholds and weights
5. Generic enough for application in multiple domains	11. Businesses do not have full information on consumer preferences	
6. Exploration of economic, social and environmental effects		

Figure 4.1: Identified key model concepts and desired model functionality

5 Related ABMS works

5.1 Introduction

This chapter builds on the identified concepts and desired functionality in the previous chapter. It examines how previous ABMS studies have incorporated such concepts and functionality, sometimes in multiple ways. The chapter maintains the distinction between general (section5.2), business-related (section 5.3) and consumer-related (section 5.4) aspects. For each category of modelling choices, the chapter provides a discussion of what choices may be best suitable for the SPREE model, given the specifications from the previous chapter. It thereby provides a flash-forward to the more elaborate description of modelling choices provided in the next chapter, identifying how those choices are embedded in literature.

Section 5.5 then specifically discusses possible frameworks for decision making processes of businesses and consumers. Section 5.6 investigates some of the more elaborate practice-oriented models that have been published, focusing on the choices made with respect to aspects discussed in this chapter. Finally, section 5.7 identifies aspects of agent-based artificial markets that are currently relatively unexplored. Again, each of these sections provides a discussion on ideas and lessons that can be incorporated in the SPREE model.

5.2 General modelling aspects

As figure 1.2 already illustrated, ABMS is a fast growing area of research. Applicable domains include biology, epidemiology, transport, financial markets and economic systems. ABMS applied to economic systems is generally referred to as Agent-Based Computational Economics or ACE (Tesfatsion, 2002). Even within ACE, the range of applications is diverse and includes studies of asset pricing, cooperative behaviour, economic policy, institutions and social norms, market design, social networks, industrial organization and dynamics, welfare economics and diffusion of innovations (Tesfatsion, 2013).

Many of these studies involve some kind of artificial market, which will also be the core of the SPREE

model for exploring effects of servicising policy. Some of the proposed models represent some specific system of interest, such as the market for mobility or water appliances. Others focus on a specific aspect or phenomenon of interest. Some publications focus on consumers, others on firms or entrepreneurs, and yet others address both sides of the market. Every model is uniquely designed to address a specific research question. Many of them include useful approaches or relevant concepts with regard to the SPREE model on servicising.

Some models explicitly consider the impact of consumption choices on resource use (Choi et al., 2012; de Haan et al., 2009) or waste generation (Chu et al., 2009). Some models (especially the more applied ones, see section 5.6) explore policy effects and external developments. None of the reviewed models explicitly considers servicising shifts, i.e. ownership-oriented (products) versus function-oriented (services) production and consumption.

5.3 Business-related concepts

This section on business choice discusses key concepts related to the business side of artificial markets that previous ABMS works have identified and explored.

5.3.1 Heterogeneity

As a first important aspect, heterogeneity between businesses represents a major difference between published agent-based models and the more traditional equilibrium models featured in neo-classical economics (Hommes, 2006). Traditional theoretical models assume purely rational behaviour of homogeneous businesses (and consumers), leading to equilibrium prices and production. The agent-based approach instead allows for extensive differentiation between firms competing in the same market. It also allows for a more human-oriented approach. Entrepreneurs can have unique preferences and beliefs about the best strategies for long-term success, that may well be influenced by their network (Ng, 2008). Each particular brand may attract a different type of consumers. Products and services from different firms have different characteristics even if they satisfy the same basic need. This heterogeneity is a crucial ingredient for explaining important observed phenomena such as the strategic choices that businesses make, which are usually different even when competing in exactly the same markets (Ng, 2008). Every modeller of agent-based artificial markets must choose how to deal with heterogeneity - even if the choice is to have no heterogeneity at all.

5.3.2 Strategic level choices

When it comes to business choice, there are two complementary types of decisions: decisions on a strategic level and on a tactical level. Strategic level choices define a firm's business model, the type of products or services that it sells (and how) and its unique competences. Such choices reflect how the entrepreneur believes it can best compete with other firms. Revisions of a firm's strategy often involve large investments of capital, time and effort.

Any servicising shift will involve altered strategic choices by businesses. One important strategic choice for all agent-based artificial markets concerns product quality. Published models by Tay and Lusch (2007) and Terano and Naitoh (2004) include quality simply as a company attribute with a value on some scale (e.g. 1-10). Alternatively, product quality can be measured on a set of different aspects.

Quality is typically interpreted as a static property of products (and services). Ng (2008) and Janssen and Jager (2002) have a more dynamic interpretation of quality and allow it to improve over time, as a result of experience and innovation. This can benefit established firms but also allows firms to catch up through R&D investments. In the extensive EURACE model described by Deissenberg et al. (2008), product quality is directly related to the skills of its employees and can change over time.

Another important aspect of business strategy that also applies to a servising shift is a firm's adaptability to a changing environment. The modelling work by Tay and Lusch (2007) shows that exploitation-based strategies (with a focus on tactical optimisation) may perform best in stable conditions, while explorative strategies (allowing for more radical strategic changes) increase the firm's survival chances in times of turbulence. Katagiri et al. (2009) have designed genetic algorithms to explore how a firm can learn about consumer preferences. In their model, a trial-and-error process of incremental change helps such firms develop products that more closely meet consumer demand.

5.3.3 Tactical level choices

Tactical level choices can be considered as the optimisation (i.e., profit maximisation) within the strategic choices, such as choosing the right price and production level. Tactical level choices can be altered more easily and more frequently than their strategic counterparts. The choices on this level directly influence price levels and market shares.

When looking at published ABMS research, there are different approaches to price and production optimisation. In the model by Chang et al. (2008), businesses set their prices according to three possible rules: just fixed prices for certain products, a fixed margin over the production costs, or a randomly set price.

Other models allow prices to adjust in response to market dynamics. In the model by Ogibayashi and

Takashima (2009), businesses adjust their price up or down by 5%, depending on their recent success. Itoh et al. (2006),Liu et al. (2010) and Tay and Lusch (2007) use genetic algorithms to let businesses learn, dynamically, what the best price setting is. Using yet another approach, Bernardino and Araújo (2013) allows business agents to set introduction prices of new goods based on market research. After that, prices slowly decrease as competing products are introduced to the market.

5.3.4 Discussion

The SPREE model focuses on producer and consumer choices relating to servicising. It is recognised within SPREE that although all business actors ultimately want to make long-term profits, they have different motivations to possibly adopt servicising. This indicates that heterogeneity between them is an important aspect, as it is in several of the studies discussed above. A reasonable approach for SPREE may be to distinguish a set of traits of products and services, all of which represent a certain quality aspect. Each business agent can then focus on a specific set of traits, of which it believes it will bring long-term success, and make strategic choices for fitting business models.

In order to be adaptable to change, business agents should regularly consider alternative business models (exploration-based). However, they should also engage in tactical level optimisation (exploitation-based), continuously optimising their selling price and production volumes. SPREE experts recognise that much real-world price optimisation is based on forecasts and market research. The market research leads to insight into expected consumer acceptance, which is then taken into account during strategic decision-making. Such considerations and activities should be reflected in the way that business agents make strategic and tactical choices in the SPREE model. Except for the model proposed by Bernardino and Araújo (2013), none of the reviewed models include autonomous forecasting and market research activities by business agents, so this would be a novel aspect.

Also largely unaddressed by examined works are the concepts of skills and infrastructure as preconditions for business models. Given that chapter 2 identifies these aspects as important possible obstacles, the SPREE generic model should somehow include these preconditions as relevant aspects of business choice.

5.4 Consumer-related concepts

As is clear from the above, the business side of artificial markets already involves tough modelling choices with several options. The consumer side, however, involves even a greater variety of possible model concepts.

5.4.1 Consumer subgroups

One of the core modelling choices regards heterogeneity, which was also identified as an important aspects for businesses. Consumer heterogeneity concerns the distribution of heterogeneous properties (or 'attributes') among the study population.

Consuming agents can represent households (Deissenberg et al., 2008) or individuals (Said et al., 2002), and their attributes can vary individually (Janssen and Jager, 2002) and/or in groups (Neri, 2007). A typical choice made in published models is to create subgroups based on common socio-economic properties such as income level and household composition (e.g. de Haan et al. (2009)). An alternative is to define clusters based on relative sensitivity to price, quality and/or other aspects (Terano and Naitoh, 2004; Neri, 2007). In domain-specific applications, the specific need may also be a governing factor in the delineation of groups. For instance, the purpose of car driving in a model about mobility-related choices (Choi et al., 2012).

The number of groups in the model may vary from just two or three (Maya Sopha et al., 2011) to forty (de Haan et al., 2009) or consumers may not be put into groups at all (Said et al., 2002). Agent preferences can be perfectly correlated with the group that they are part of (Terano and Naitoh, 2004) or randomly distributed over agents from all socio-economic subgroups (de Haan et al., 2009). The modeller's selection of consumer subgroups and the related assignment of agent attributes clearly involve many degrees of freedom and should be chosen carefully to fit the problem being studied.

5.4.2 Personal preferences

Strongly related to the subdivision of the study population into groups, the modeller has to decide on what basis the consumer agents will choose between offers on the market. An important distinction here is between personal preferences (or 'personality traits' (Said et al., 2002)) and social influences (Janssen and Jager, 2002; Maya Sopha et al., 2011). The first important preference, or constraint, in many studies is the agent's budget or disposable income (e.g. Eppstein et al. (2011), Choi et al. (2012)). Product quality (also discussed earlier in a business context) is the second important aspect and used almost interchangeably with extracted 'utility'. Quality (or utility) can be expressed simply on a scale (0-1) (Laciana and Oteiza-Aguirre, 2013; Bernardino and Araújo, 2013; Eppstein et al., 2011). Alternatively, (subjective) quality can be represented as a set of relevant attributes that consumers have preferences for. In that case, products and services have a score on each of the attributes (Choi et al., 2012). Consumers may have different sensitivity for both price and quality (Terano and Naitoh, 2004; Zhang and Zhang, 2007).

Moving on to the concept of personality traits, consumer attitudes may represent varying levels of innovativeness (Schwarz and Ernst, 2009) (Said et al., 2002), opportunism (Said et al., 2002) and materialism

(Maya Sopha et al., 2011). Uncertainty and dissatisfaction with products may lead to agents reconsidering their favourite vendor (Janssen and Jager, 2002). On the other hand, loyalty can be an important concept that ties consumers to their current provider of products or services (Chang et al., 2008; Itoh et al., 2006; Kirman, 2008)

5.4.3 Social influences

Social influences can change personal preferences or even constitute the primary basis for consumer choices. An important concept in this context regards network structures. Network structures among consumers represent an intensively studied subject. Many papers on artificial markets include explicit assumptions on social networks, usually in the form of small-world networks (Deissenberg et al., 2008; Eppstein et al., 2011; Janssen and Jager, 2002) (Laciana and Oteiza-Aguirre, 2013; Neri, 2007). These social networks may affect consumer behaviour in different ways. Typically, consumers will somehow conform their preferences to social contacts (Eppstein et al., 2011; Itoh et al., 2006; Laciana and Oteiza-Aguirre, 2013; Terano and Naitoh, 2004) or directly imitate each other's consumption behaviour (Said et al., 2002; Janssen and Jager, 2002). As a variety, the social network may limit the number of alternatives that a consuming agent can consider (Neri, 2007). Yet another possibility involves 'positional' considerations, meaning that consumers determine the subjective value of a product's 'utility' *relative to* consumption choices of others in their network (Bernardino and Araújo, 2013).

5.4.4 Bounded rationality, subjectivity and learning

Although many types of personal preferences and social influences may affect consumer (and to some extent, business) choice, there is only a limited amount of information that consumers can process and take into account for making any decision (Neri, 2007; Sasaki et al., 2011). This phenomenon is known as bounded rationality (de Haan et al., 2009). Making matters worse, consumers are subject to blurred perceptions (Laciana and Oteiza-Aguirre, 2013; Ohori and Takahashi, 2012). Bounded rationality and subjectivity may lead to irrational choices and have attracted the interest of many scholars. ABMS is a perfectly fitting tool to specifically study such effects or take them into account in the context of a more extensive model.

Package descriptions (Neri, 2007), product image and advertising (Neri, 2007; Terano and Naitoh, 2004) and media influences (Eppstein et al., 2011) may all affect consumer's subjective perception of offers on the market. A possible method to deal with this subjectivity in a computational way is to use 'fuzzy logic', allowing the agent to process complex information in a way that more closely captures the concept of bounded rationality (Tay and Lusch, 2007).

Despite the presence of bounded rationality an subjectivity, consumers are able to learn (over time) about the options they are presented with. For instance, Liu et al. (2010) and Ohori and Takahashi (2012) study consumer agents' learning behaviour in the context of artificial markets. Their respective models feature genetic algorithms to allow agents to learn incrementally about the merits of various consumption models.

5.4.5 Discussion

For the SPREE generic model, the previous chapter already identified that heterogeneity between consumers is an import aspect in explaining why some consumers adopt service-based consumption models more quickly than other consumers. The published models discussed above provide a good basis for making modelling choices with regard to heterogeneity in the SPREE model. For instance, similar to what section 5.3.4 described for business agents, consumer agents in the SPREE model can have unique preferences for traits of products and services that relate to specific quality aspects. This means that they perceive quality (of products and services) as a multi-attribute construct.

Being a generic model, the SPREE simulation model should allow for heterogeneity in lifestyle aspects between and within groups of consumers. The specific identification of consumer groups may vary per case study but could represent, for instance, different socio-economic groups. Consumers should be defined generically enough to represent either individuals, households, consuming businesses or whatever representation is most suitable for the specific case study. The number of groups should also be freely adjustable to the specific case study.

Quality of life is an important social objective and can be understood in two ways: the perceived quality of fulfilling the central consumption need in the model, but also the possibility of spending saved money on other activities or functions that may enhance quality of life. The former interpretation can be measured by looking at the fit between consumer preferences and their chosen consumption model. The latter interpretation, which is also known as 'rebound effects', is not explicitly included in the model. However, saved money can be estimated from model outcomes by comparing total consumption costs in the current situation and total costs in a new, possibly servicised situation.

In their evaluation of possible consumption models, consuming agents should compare offers from business agents based on their respective scores on each of the quality-related preferences. In addition, consuming agents should somehow be sensitive to social influences. Ideally, these influences would be based on social network structures, but this was found not to be a realistic ambition within the scope of SPREE, especially with regard to required data collection and structural variety between case studies. As a compromise, consumer agents in the SPREE generic model may consider the overall market share of each type of con-

sumption model as a proxy for social influences. Also, consuming agents may take into account their own historic choices, which may lead to the repetitive behaviour that is assumed by some of the models discussed above.

The referenced above suggest that bounded rationality, subjectivity and learning clearly affect consumer choice. They are therefore potentially important concepts in an agent-based model of an artificial market. However, they are not included in the SPREE model. A limited number of aspects could be included, and SPREE experts considered other factors to be more important determinants of consumer choice between products and services.

5.5 Decision-making frameworks

5.5.1 A variety of frameworks

The sections above already identify a number of factors that may influence decisions of producers and consumers in a given market. In order to understand the whole decision-making processes, the modeller also needs to identify how the factors are weighed together by market participants when deciding between different options.

Decision-making frameworks generally involve two elements:

- A method for multi-criteria analysis (MCDA): how are multiple 'scores' combined into one (or a limited set of) numerical value(s) to be compared between options?
- A comparison structure: given the resulting value(s) from the MCDA, how do decision makers decide which option is best?

Past studies propose such a variety of possible decision-making frameworks that there clearly is not one single generally accepted framework. Kiesling et al. (2012) identifies this area as one of the aspects of ABMS that is still in an early stage of development. The sections below compare the approaches of several published agent-based models.

5.5.2 Multi-criteria analysis

When presented with different options, agents may simply rely on basic heuristics to help them choose. Such heuristics include imitation, repetition and 'social comparison', copying either their own previous choice or the choice made by someone in their social network (Janssen and Jager, 2002; Maya Sopha et al., 2011).

More 'rational' agents will make a more deliberate choice. They may assign some kind of numerical value, or values, to every relevant option and base their decision on that value. Again, there are gradations in the level of sophistication.

- Lexicographic matching is one of the simplest approaches. In this approach, every consuming agent has a 'favourite' product attribute and will always choose the product (or company, or combination) that scores best on that attribute, no matter the scores on other attributes (Karacapilidis and Moraïtis, 2001; Schwarz and Ernst, 2009).
- With the additive utility approach, agents put a weight on every attribute and then sum the (normalised) weighted scores to obtain a total score (Chappin et al., 2007; Afman et al., 2010).
- As a variation to additive utility, consuming agents may have target levels for product attribute values, describing their 'ideal' product. The sum of the squared deviations from these target levels represents how close it is to the agent's ideal product (Neri, 2007).

5.5.3 Comparison structures

Even with numerical values assigned to every relevant option the consumer is presented with, there are multiple ways to make a choice among them.

The most obvious possibility is to simply select the option with the highest score (e.g. Schwarz and Ernst (2009)). A variation to this is the use of pairwise comparisons, where product scores (e.g. one for cost and one for quality) are valued relative to other products (Itoh et al., 2006; Eppstein et al., 2011).

Alternatively, decision makers assign a probability based on the obtained total utility of each option. This approach is known as discrete choice modelling and has multiple variants, of which two are listed below:

- The most straighforward approach is to assign probabilities that are directly proportional to the obtained utilities (Katagiri 2009). However, this is not necessarily realistic: intuitively, the best option will typically have a more than proportional probability of being selected.
- A more sophisticated variant is called 'multinomial logit' (de Haan et al., 2009; Deissenberg et al., 2008). Multinomial logit models assume logarithmic relations between utilities and choice probabilities, effectively increasing the likelihood that the option with the highest utility is chosen.

One of the most significant problems with the mentioned probability-based approaches is that they necessarily assume independence of irrelevant alternatives (IIA). IIA means that the relative probability to choose between two alternatives is independent of the presence of a third alternative. So, if a market has

offers for two different products A and B, and a product C would be introduced, the probability that a given consumer would choose B over A should remain the same (although the total probability of choosing either decreases, as product C also gets a share of the total probability). This is a problematic assumption in the case that A and C have some overlapping characteristics and are therefore substitutes for each other. In that case, the introduction of product C would probably affect the probability of choosing similar product A more than the probability of choosing distinctive product B. This violates the assumption of IIA, where the introduction of any 'irrelevant' product C should not affect the relative probability of choosing B over A. The presence of (partial) substitutes is a real phenomenon in many markets, making the probability-based approaches unreliable to use unless the subset of options is chosen very carefully.

Somewhat related to the IIA problem, Zhang and Zhang (2007) use ABMS to explore the 'decoy effect' that seems to affect real-life consumption processes. The decoy effect relates to the phenomenon that the mere presence of a third option affects the relative perceived utility of two other options, clearly violating IIA assumptions. For instance, imagine a marketplace with two options: option A which is cheap and has low quality, and option B, which is more expensive but also offers higher quality. The market has one consumer that considers both options equally favourable, meaning that the additional cost of offer B is exactly equal to the consumer's willigness to pay for the additional quality. Now a third option is introduced, which is even cheaper and has an even lower quality than offer A. Scientific experiments show that such an offer *increases* the relative preference for offer A, which is no longer the cheapest option (Zhang and Zhang, 2007). A possible explanation is that consumers implicitly consider the average or median scores on product attributes when making a choice, and may not typically want to deviate much from these scores.

As a final note on comparison structures, a possible pre-processing step that can be combined with any of the discussed approaches could be a filtering process based on attribute (and budget) threshold values. The agent can then choose one of the remaining options at random (Choi et al., 2012) or use one of the frameworks outlined above.

5.5.4 Discussion

As already suggested above, products and services in the SPREE model have 'scores' on a variety of traits that distinguish them from competing products and services. Producing and consuming agents compare these scores with their own set of preference weights and thresholds. They obtain a total utility as the sum of multiplied weights and scores (following the additive utility approach). With regard to the comparison framework, we have chosen for deterministic pairwise comparison, meaning that agents will always consider options in pairs, always choosing with 100% certainty the option that comes out best. For a number of

reasons, we think this approach is more suitable than a probability-based approach:

- Deterministic pairwise comparison does not rely on the IIA assumption. This is good, because IIA puts severe constraints on case study parametrisation by not allowing any overlap between alternatives. In certain case studies, options with overlap are difficult or impossible to avoid, so IIA would be a problem.
- It does not require the user to derive formula's that relate perceived utility to choice probabilities.
- Resulting choices by agents are easier to interpret, validate and communicate.

When comparing two choices, agents in the SPREE model will evaluate the difference in total perceived utility *relative to the difference in cost*. This means that the total assigned net utility of each option (i.e., combination of costs and utility) does not take an absolute value, but is always measured relatively to alternative options. This approach captures some of the subjectivity identified as important for decision-making and also partially addresses the decoy effect.

To illustrate how the pairwise comparison approach deals with the decoy effect, consider the following example. A consumer can choose between two products A and B. Product A has a price tag of 150 euro and a perfect score (i.e., 10 out of 10) on the relevant preference traits. Product B costs 100 euro and scores 7.5 out of 10. The consumer is willing to pay 20% more for a one-point higher preference score. The score difference is 2.5, which is worth 2.5 * 20% * 100 = 50 euro to the consumer. This makes both products equally attractive to this consumer.

Product	Price	Score
A	150	10
B	100	7.5
C	75	5

Table 5.1: Parametrisation for the example on the decoy effect

Now consider the introduction of product C, with a price tag of 75 euro and an overall score of 5. When comparing A and C, the score difference is 5. The consumer is thus willing to pay 5 * 20% * 75 = 75 euro more for product A, making A and C equally attractive. However, when comparing B and C, the score difference of 2.5 results in a willingness to pay of 2.5 * 20% * 75 = 37.5 euro for product B relative to product C. Since product B only costs 25 euro more, the consumer will clearly favor product B over C! With all other pairwise comparisons equal, this means that option B is now clearly the best choice for the consumer, simply because of the introduction of inferior product C. The SPREE model is thus able to (at least partially) capture the important psychological decoy effect.

With regard to the notion on pre-processing of alternatives, this can be useful for the SPREE model to incorporate budget and preference value thresholds for producers and consumers. A filtering step is therefore included in the decision frameworks within the SPREE generic model.

5.6 Concrete ABMS applications

This section discusses some concrete applications of ABMS artificial markets that have been developed to inform policy. It highlights the choices made on the business side and the consumer side of the model and the most significant outcomes. The examples illustrate how previous studies combine several of the components outlined in this chapter and how ABMS can provide valuable input for policy.

5.6.1 Car purchases in Switzerland

Mueller and de Haan (2009) propose a large-scale (100,000 household agents) model of car purchase choice in Switzerland. The households are subdivided into 40 socio-economic groups. Car attribute preferences take the form of thresholds and are homogeneously distributed over the groups. The model involves a database of several thousands of possible cars, but consumers only consider a limited number of options at once, partly based on the cars visible within their social network. They either use heuristics or deliberation (according to multinomial logit) to make a choice. The model outcomes suggest that proposed 'feebate' (a self-financing system of fees and rebates) policy induces consumers to switch to cars with more efficient engines, but not to different vehicle classes. Also, the results suggest that the policy has limited rebound effects such as the purchase of more or bigger cars or more miles driven.

5.6.2 Social influences on mobility choices

Eppstein et al. (2011) proposes another model on mobility choice, focusing on social or psychological factors and 'neighbor effects' on consumer attitudes. Consumers have socio-economic attributes, a social network or neighbourhood, a market penetration threshold and a level of 'rationality' that indicates their susceptibility to social and media influences. In addition, consumers have a 'G'-factor that indicates to what extent the consumer overestimates certain benefits of fuel-efficient cars. The 'G'-factor is a similar concept as ' willingness to pay', as it is a modifier that helps make the trade-off between cost and utility. The g-factor is assumed to change over time due to influences by social networks as well as the media. Decision-making processes involve pairwise comparisons of the offers that meet certain thresholds with regard to budget and market penetration. The model outcomes suggest that "PHEV market penetration could be enhanced

significantly by providing consumers with ready estimates of expected lifetime fuel costs associated with different vehicles (e.g. on vehicle stickers), and that increases in gasoline prices could non-linearly magnify the impact on fleet efficiency."

5.6.3 HEV adoption in Korea

Choi et al. (2012) also proposes a mobility choice model, this time for 10,000 individual Korean consumers, with a focus on the adoption of hybrid electric vehicles (HEVs). Consumers have individually varying income statement, monthly budget for mobility (fixed and variable), purpose of drive, and vehicle preferences for size, oil type, fuel economy and CO2-emissions. The study does not include an explicit social network. The simulation results suggest that current policy can lead to a penetration rate of HEVs of 6.4% over a time span of 20 years. Price decreases or additional subsidies within plausible ranges may further increase the penetration rate to around 11.5%.

5.6.4 Water devices in Beijing

Chu et al. (2009) focuses on the adoption of water appliances in Beijing, with the eventual societal goal of reducing residential water use. Households constitute the primary agents in the model. The model distinguishes eight types of end-use functions and five types of water devices that provide these functions. Households have individually varying needs for the 8 functions. They use three possible decision rules for selecting new appliances: habitual/repetitive, random/indiscriminate, and deliberate (i.e., minimising total cost). The study investigates the effects of various policies aimed at decreasing freshwater use. It suggests that the replacement (coordinated by the government) of faucets and showers can readily make improvements of water use efficiency at a very low cost, while the replacement of low-efficiency laundry machines shows minor efficiency improvements in water use efficiency at a relatively high cost.

5.6.5 EURACE: a massive model of coupled markets

The final model discussed in this section is the EURACE model (Deissenberg et al., 2008). EURACE is "a major European attempt to construct an agent-based model of the European economy with a very large population of autonomous, purposive agents interacting in a complicated economic environment." The model being developed (it is still under construction) is indeed massive and involves newly developed modelling frameworks that allow the interaction of submodels running in parallel. The integrated model simulates the interactions on and between five relevant markets, some on the level of European economically coherent NUTS-2 regions and some on a 'global' level: consumption goods (local per region), investments goods

(global), labour (local), credit (local) and financial assets (global). It involves inter-agent relations such as buyer-seller, firms-worker and bank-firm. The model features the use of real GIS data to account for geographical differences. In addition, it assumes various network structures between agents. Producers set prices, demand labour, produce sellable goods and obtain technology. The quality of their goods relates directly to the skills of their employees. Consumption choices (by households with socio-ecocomic properties) are based on prices and quality, with probabilities generated through multinomial logit. Initial model outcomes show that the model is capable, for instance, to reproduce realistic relations between wages, skills and unemployment with certain variations that traditional economic models cannot properly account for.

5.6.6 Discussion

The examples above identify some ABMS studies that are relatively similar to the SPREE model. They illustrate alternative approaches of combining some of the possible modelling choices discussed in this chapter into an agent-based model. The examples especially help understand how the SPREE model differs from other proposed models and thereby contributes to development of the ABMS field. The two primary differences with these models (except for EURACE) is that SPREE is of a generic nature instead of being specifically designed for a specific domain, and that the SPREE model considers choices of both consumers and producers. When comparing with EURACE, SPREE allows for much more detail on behavioural considerations and supports a greater variety in product and service traits. Moreover, EURACE consists of five coupled submodels, while the SPREE model integrates everything in just one model. Another unique feature compared to all these models is that the SPREE model explicitly enables the study of servicising shifts, which - to our knowledge - no previous studies address in such a comprehensive way.

5.7 Future research

5.7.1 KISS: simple models, only extended when needed

According to Kiesling et al. (2012), there are two fundamentally different approaches to ABMS model development. The KISS approach (advocated, for example, by Kirman (2008)) advises to 'Keep It Simple, Stupid!'. This approach stresses that models should simplify reality as much as possible and exclude all elements that are not absolutely necessary to study the emergent pattern of interest. Additional complexity should only be added if the original set of inputs cannot explain the pattern. The KISS approach results in elegant, abstract models that reduce computational requirements and make simulation outcomes easier to interpret and communicate. This makes the KISS approach very suitable to study theoretical phenomena

or mechanisms, such as the effect of network structure on some diffusion process. KISS-based models are great for showing how simple system structures and rules can lead to complex outcomes that also have counterparts in reality.

5.7.2 KIDS: descriptive models, only simplified where justified

Many real-world systems and phenomena, however, are the result of *several* complex and interrelating mechanisms and decision rules. The dynamic interaction of effects leads to different results than just the sum of individual effects. Modelling such complex systems demands more complex models. Edmonds and Moss (2005) therefore reverse the KISS approach and advocate a KIDS approach: Keep It Descriptive, Stupid! In the KIDS approach, modellers are urged to keep the model as descriptive as possible and capture system elements, relations and behaviour from reality as straightforwardly as possible. Simplifications should only be made where justified. KIDS models are especially useful for ABMS models tailored to particular domains. The close fit to reality makes agent decision rules easier to parametrise with questionnaires during data collection (Zenobia et al., 2009). KIDS models provide managerial guidance and inform policy analysis on practical problems. However, the descriptive nature only really works out in specific domains, otherwise they would no longer be truly descriptive of some specific system (see also the examples in 5.6). As a result, most KIDS models are not generic enough for use outside their intended domain (Kiesling et al., 2012).

5.7.3 A need for models that are both complete and generic

Kiesling therefore identifies the large gap between KISS and KIDS models as 'the area in which progress would be most beneficial in terms of providing managers with simple, robust, adaptive and easy to control models that are as complete as possible and still applicable to a range of applications as wide as possible. (...) Both more solid empirical foundations and better, more adaptable and versatile models need to be developed' (Kiesling et al., 2012). A modular approach, where parts of the model can be easily replaced by alternative options (e.g. decision-making heuristics), further increases the practical value of an agent-based artificial market model (Zenobia et al., 2009).

5.7.4 Other promising directions

Despite all the modelling work discussed in this chapter, certain concepts appear still to be relatively unexplored. Some of these concepts may be particularly relevant to servicising, especially to the business side of it. One concept that appears not to have received much attention in the considered literature is the conceptualisation of the general structure (other than some characterising aspects) of business models. In

the context of artificial markets, business models that describe product-service systems remain especially unaddressed. Also, none of the explored models involve the physical flow of materials through business and consumption models, making it difficult to derive any quantitative effects on resource use and environmental impacts. Other literature (e.g. Davis et al. (2008)) addresses such flows but does not include any of the behavioural aspects that are so important in artificial markets. Finally, the type of decisions that business agents make remain extremely simplified versions of real world business decisions that involve, for instance, detailed forecast analysis, cost-benefit analysis and strategy (re-)evaluation.

5.8 Conclusion

This chapter has reviewed several aspects of published studies on artificial markets that may in some way be applicable to the SPREE model on servicising policy. The studies show various ways of dealing with heterogeneity among consumers and producers. They also cover a wide variety of decision-making routines of modelled agents, on multiple levels and based on various indicators of extracted utility. The literature suggests that bounded rationality and subjectivity make choices less rational (and predictable), while learning processes can help agents to make better choices over time.

ABMS already has several applications for informing real-world policy. These models combine several of the outlined aspects in interesting ways, resulting in tailor-made models for the problem at hand. An evaluation of promising directions for future agent-based models reveals that future artificial market models should be more descriptive (as opposed to simplistic), that they should consist of generic, modular elements and that current models remain very limited in their representation of business models and strategic choices by business agents.

Figure 5.1 provides an overview of the modelling choices discussed in this chapter, and the suggested choices for the SPREE generic model. The figure does not imply that if a model would include all the listed elements, it would be perfect and capture all aspects that can possibly be relevant to agent-based artificial markets. Nonetheless, the overview puts the choices made into the perspective of other possibilities. The marked choices are worked out in more detail in the next chapter.

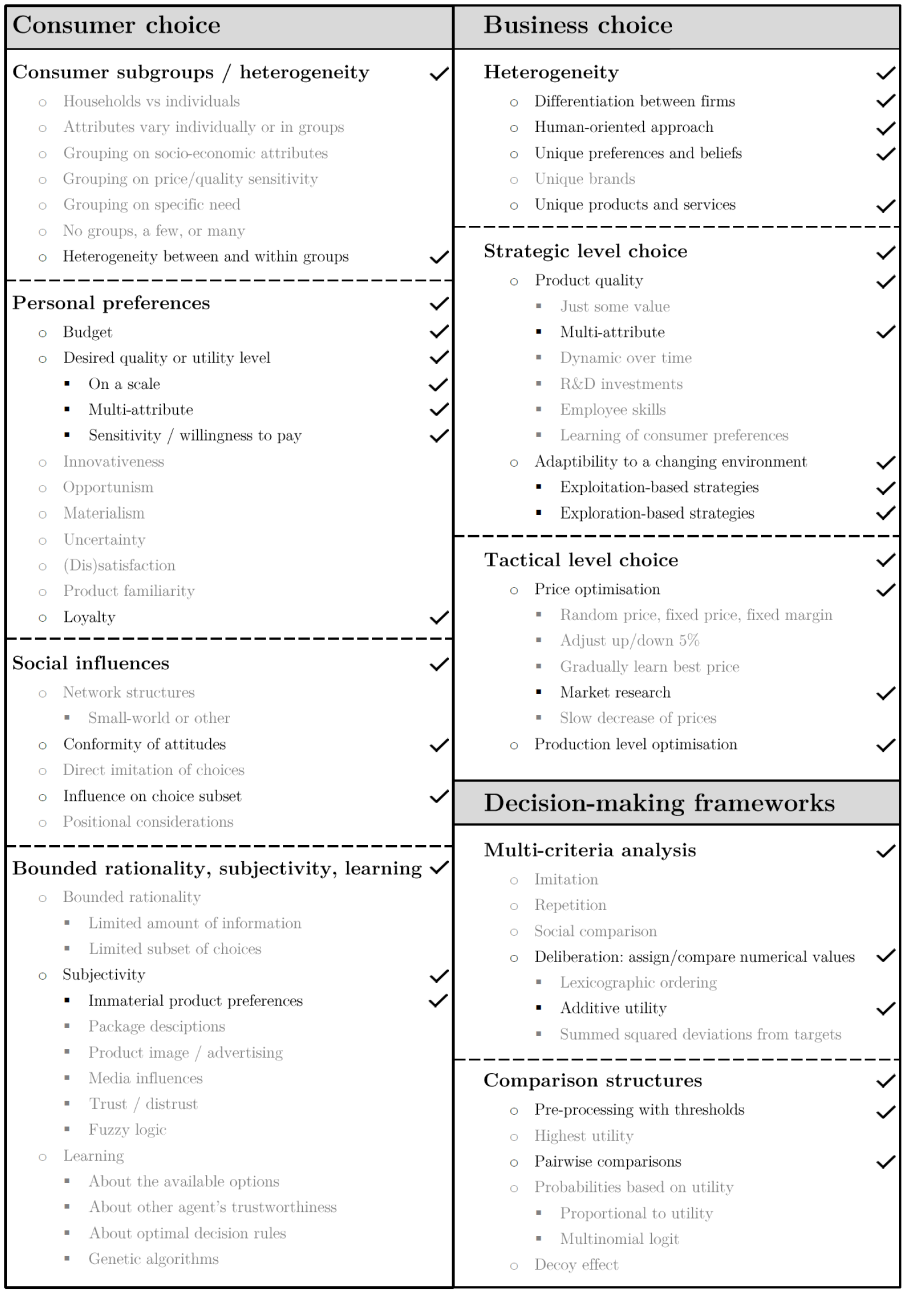

Figure 5.1: A non-exhaustive list of identified modelling options from review literature, with suggested modelling choices for SPREE marked with a 'V'

6 Model description

6.1 Introduction

This chapter presents the elements of the SPREE generic model, how they interrelate and how they behave in the model. It builds on the identified concepts and desired functionality from chapter 4 and the discussion of literature in chapter 5.

Section 6.2 discusses the comprehensive/holistic approach that characterises the SPREE generic model development. Section 6.3 highlights the scope and general structure of the model. Section 6.4 discusses important business-related aspects and section 6.5 the consumer-related aspects. Section 6.6 identifies some important limitations of the model, that follow from the choices made.

The information in this chapter is a summarised version of the much more elaborate information in appendices B and C. Appendix B discusses in more detail the 'what' of the model. It formalises the described characteristics of all agents and objects in the model as formal agent/object states and behaviour. Appendix C elaborates on the 'how' of the model. It describes in detail the structure and content of important model procedures, including decision-making algorithms for the agents' strategic and tactical choices.

6.2 General approach

6.2.1 Addressing identified research gaps

The SPREE generic model aims to explore servicising shifts and support the development of servicising policy packages. Given its position within a large European project, the model development provides a great opportunity to address some of the identified 'gaps' (or promising directions) in ABMS research identified in section 5.7. In particular, the SPREE generic model directly addresses the call for models that are both generic and complete.

6.2.2 A descriptive approach

The model development will follow a primarily KIDS-based approach (see section 5.7). The large number of relevant and interrelating concepts identified as important in chapter 4 already shows that the model can never be made as simple as the KISS approach prescribes. Such simplicity and elegance facilitates analysis and communication of results, but would imply that too many concepts must be discarded or remain harsh abstractions of reality.

The straightforward, comprehensive approach that KIDS prescribes much better fits the identified complexity and allows to communicate about the model in real-world terminology instead of abstractions. Satisfying the KIDS approach, the model has been formulated as descriptively and elaborately as the available timeframe allowed. A problem here is the identification of a specific real-world system that the model should represent. The whole idea of the SPREE generic model is that it does not yet represent a specific market but is instead applicable to many sectors and domains. During model development, a main challenge was therefore to formulate model elements, relations and behaviour such that they are simultaneously descriptive and generally applicable.

6.2.3 A truly comprehensive model

Some of the identified concepts are only relevant in some or even one sector(s). For example, the presence of 'skills' can be very relevant in the agri-food sector, as specialised businesses can be more effective and efficient in combatting pests than individual farmers (although some farmers are very skilled as well). In contrast, the concept of skill barely plays a role in servicising shifts in the mobility sector: the operation of owned and shared cars requires the same basic skills. The SPREE generic model includes concepts such as skill (and other concepts identified in chapter 4) in a modular way that makes their use and the associated input data optional. For instance, agents are programmed to be as 'smart' as possible, taking into account a theoretically infinite set of considerations, but the model also includes possibilities to show increasingly non-rational behaviour.

With this approach, the same model can be used for all three case studies. Any concepts that are not considered relevant in a specific case study, or cannot be adequately supported with a theoretical basis or empirical data, can still be switched off in the model. These configuration options make the model flexible in its applicability. However, they also imply that results will not be directly comparable between case studies as they are based on different 'subsets' of the model.

6.2.4 Benefits of a descriptive and comprehensive approach

The comprehensive KIDS-based approach means that the SPREE generic model includes more functionality than would be required for either of the three individual case studies in the SPREE project. The development of the generic model takes more time and effort than would have been needed in the alternative process of including into the generic model only those elements that are certain to be present in all three case studies. In this alternative, more KISS-like, approach the generic model can be extended where needed, resulting in three distinct, tailor-made models. A truly KISS-based approach would not even bother with a generic version, but simply design three separate models, possibly reusing some parts but without structural overlap.

Nonetheless, the comprehensive KIDS-based approach offers a number of benefits that justify the allocation of more time and effort to the generic model development. First, including functionality in earlier stages of model development takes less time and effort than in steps closer towards implementation, as the number of dependencies and the level of detail are both relatively low. The main reason for this difference is that additional functionality always relates in some way to existing functionality, and virtually always requires (expensive) changes to the original model code. This rework is unnecessary if the respective functionality is included from the start.

In addition, significant synergies may result from a comprehensive approach that are nonexistent when developing three largely separate models. In practice, some revisions of the generic model after its official delivery date are unavoidable, typically resulting from new insights obtained during model experimentation. Having to implement the needed changes only once instead of three times saves a significant amount of time, making up for the additional time investment up front.

As a final benefit, the comprehensive KIDS-based approach facilitates future use of the model as a substantive basis for applications in more sectors and new studies of artificial markets.

6.3 Scope and basic structure of the model

6.3.1 Overview

The high-level modelling choices that characterise the SPREE generic model reflect the aim to be complete, yet generic in the modelling approach. This section (6.3) defines the scope and structure of the system that the model will represent, in terms as specific as possible given the desired generic applicability. The two following sections (6.4 and 6.5) characterise in some more detail the business and consumer agents and the decisions they make in the model. Figure 6.1 shows for each of the three sections which concepts from

chapter 4 it addresses.

6.3: Scope and structure	Product	Service	Consumer need
	Business actor	Consumer actor	Competition
	Environmental impact	Resource use	Generated waste
	Policy instruments	Policy effects	External developments
6.4: Producing Businesses	Business model	Consumer acceptance	Business profits
	Skill	Innovation	Infrastructure
	Ownership-oriented	Function-oriented	
6.5: Consuming agents	Consumption model	Consumer spending	Social objectives
	Lifestyle	Quality of life	Loyalty
	Ownership-oriented	Function-oriented	

Figure 6.1: Concepts identified in chapter 4, and where they are addressed in sections 6.2high-level choices

6.3.2 Model elements

Box 6.2 provides a list of the elements that the SPREE generic model consists of. It completes the system identification step (step 2 of the methodology). The figure distinguishes the Observer, agents, objects and links as the four high-level categories of elements. Agents represent individuals, organisations, the 'world market' and the 'physical environment'. Objects represent 'things'. Box 6.2 further makes the distinction between active and passive agents, as well as tangible and intangible objects. Active agents and tangible objects can interact with each other. Passive agents and intangible objects basically just contain information that the other agents and objects can use. The figure already contains a brief description of each model element.

6.3.3 Basic market structure

The SPREE generic model represents a generic market of sellers and buyers that revolves around some central consumption Need (e.g. mobility, use of fresh water). The consumption Need can be satisfied through a predefined set of Products and Services. The predefined set makes sure that the model stays close to reality

Observer	
Observer (O)	Does not participate in the market, but can influence it as external 'force'
Agents	
Active market participants	
Producing Business (PB)	Produces and sells Products and Services
Consuming Business (CB)	Consumes Products/Services to satisfy Need
Consuming Individual (CI)	Consumes Products/Services to satisfy Need
Passive agents	
World Market (WM)	Sells and buys Products
Physical Environment (PE)	Accepts wastes
Objects	
Tangible objects (can be `owned' by agents)	
Tool (Product) (TP)	Remaining usability measured in use time
Consumable (Product) (CP)	Remaining usability measured in quantity
Manufacturing Model (MM)	Basis for PB production
Sales Model (SM)	Results in offered Product or Service
Consumption Model (CM)	Specifies Product/Service consumption
Intangible objects (just contain information)	
Service (S)	Defines a service produced by an SM
Infrastructure (I)	Specifies a relevant infrastructure
Skill (SK)	Specifies a relevant skill
Resource (R)	Represents a resource such as steel or CO_2
Policy Package (PP)	Collection of Policy Instruments
Policy Instrument (PI)	Collection of Policy Effects
Policy Instrument Effect (PIE)	Defines a specific effect on the market
Market Development Event (MD)	Collection of Market Development Effects
Market Development Effect (MDE)	Defines a specific effect on the market
Links	
Service Contract	Connects seller and buyer of a Service

Box 6.2: List of elements of the SPREE generic model

and can be fully parametrised by the user. Box 6.2 shows that there are two types of Products, i.e. Tools, which have a functional use time, and Consumables, which are always directly consumed when used. Services are represented as Services Contracts between sellers and buyers.

Figure 6.3 illustrates that the main interaction in the model is between producing agents and consuming agents. Producing Businesses (PBs) have a 'business model' consisting of a Manufacturing Model (MM) and one or two Sales Models (SMs). The business model allows PBs to do input/output conversions and thereby offer Products and/or Services for sale on the market. The PBs compete for market share among the consuming agents (CAs), which can be represented as Consumers or Consuming Businesses - the model accounts for both B2B and B2C markets (and mixtures). Consuming Businesses have different properties and behaviour from Producing Businesses, although both are profit-oriented. All consuming agents have a

Consumption Model (CM) that defines how they can satisfy their Need through a specific Product or Service.

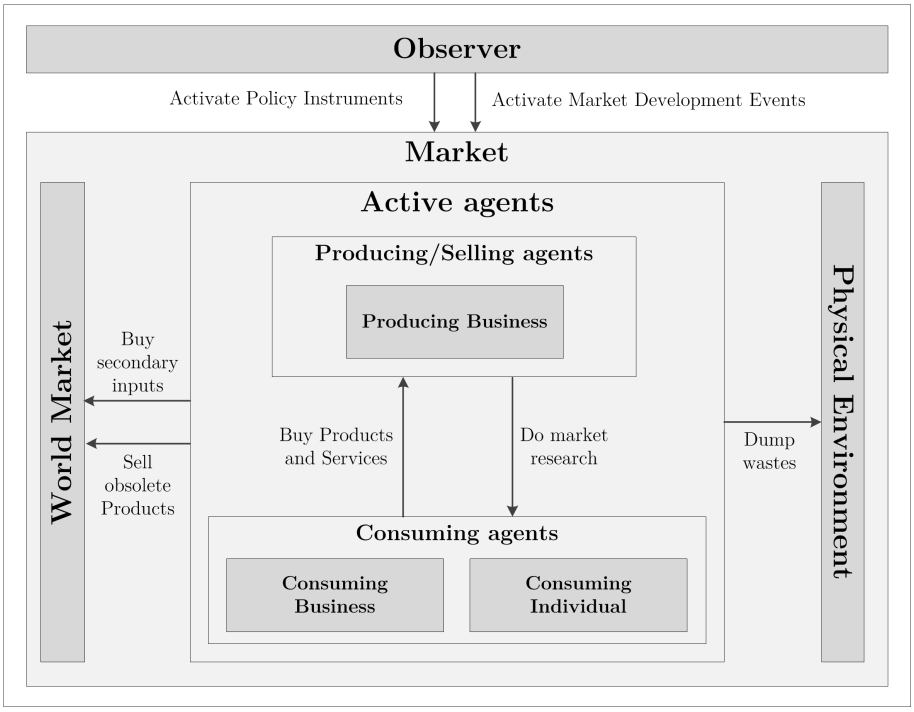

Figure 6.3: Relations between agents in the model

6.3.4 Material flows

The input/output conversions in MMs, SMs and CMs result in material flows with associated resource use and generated waste. As shown in figure 6.3, agents can buy inputs from (and sell obsolete Products to) the World Market, which is a static agent that does not participate in market dynamics in other ways than providing material inputs and accepting material outputs. Outputs that cannot be sold to the World Market have to be dumped in the Physical Environment, which is a similar type of agent as the World Market.

Figure 6.4 provides a closer look at the material flows and shows, for example, that stocks of (material) main inputs and outputs are located at SMs and CMs. The distinction between product-based and service-based SMs allows for partial and gradual servicising, where a PB can gradually allocate different amounts of production capacity to either of the SMs.

As a result of the conversions, all Products and Services in the model have associated resource use and generated wastes. These are quantified based on a predefined set of Resource types. For example, in one case study the resources of interest can be steel, a certain type of plastic, oil and CO_2. For all Products

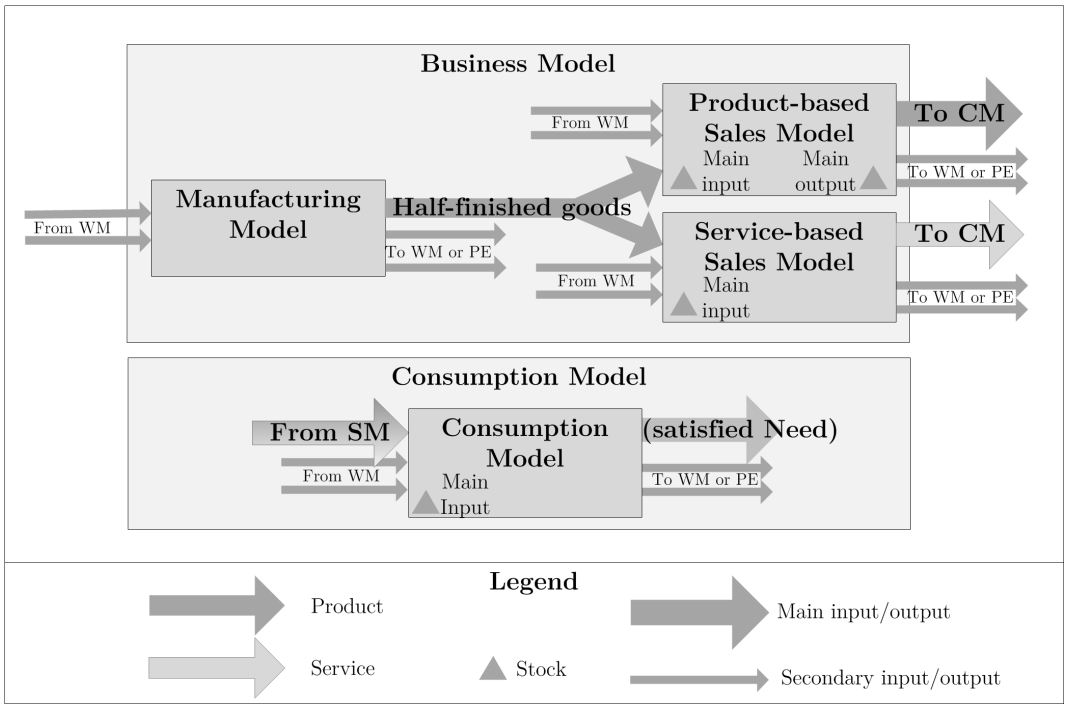

Figure 6.4: Material flows in the model

and Services, the model keeps track of the amount of each of these resources that was extracted from nature as virgin resource inputs, and how much secondary outputs had to be dumped as wastes throughout all conversion processes and at the end of life of the product.

6.3.5 External influences

Figure 6.3 also illustrates the Observer's position 'above' the market. The Observer is an auxiliary agent that can influence the market by (de)activating Policy Instruments (structured in Packages) and Market Development Events. Policy Instruments and Market Development Events have Effects that may theoretically influence all properties of agents and objects in the model. Policy Instruments represent actions that the problem owner can take to influence the market, such as providing a subsidy or banning a certain Product. Market Developments represent completely external changes that affect the market, such as resource price fluctuations and scientific breakthroughs. Together, the Policy Instruments and Market Developments allow for the exploration of several policy options in multiple scenarios of future developments on the market.

6.3.6 Time unit

The basic time unit in the model can be adjusted based on the specific market it represents. The model works equally well with time units of a day or a year since none of the model elements have a direct relation to a specific timespan. Therefore, it is up to the user to ensure that all time-related input data are formulated consistently.

6.4 Producing Businesses

6.4.1 General characterisation

Producing Businesses (PBs) are characterised by their strategic beliefs and by their initial Business Model. The strategic beliefs represent that different entrepreneurs have different ideas on how to 'conquer' a market. Some PBs may focus on offering products or services that have certain quality aspects (e.g. user-friendliness or robustness). Others may have an environmental focus. Yet others may just try to offer Products that 'do the trick' at the lowest cost (and selling price) possible. This characterisation of PBs represents a human-oriented approach to businesses and accounts for the heterogeneity that leads to the variety of offered Products and Services observed in reality.

6.4.2 Business model

Business models are represented as a chain of two interrelated transformation processes: the Manufacturing Model (MM) and the Sales Model (SM). This is still a very abstract representation of reality: section 6.6.4 discusses some aspects of business models that the SPREE generic model excludes.

The MM represents how the PB obtains the material product that forms the material basis (if any) for what they sell on the market. They may just buy it directly from the World Market or manufacture it themselves based on inputs bought on the World Market. The MM can even be empty in case the PB offers a Service that is not based on any material product. In addition to an MM, the PB can have up to two SMs: a product-based SM and a service-based SM. If the PB operates two SMs simultaneously, they must both use the outputs of the same MM. PBs produce stocks up to target stock levels, with a daily production volume limited by their chosen capacity.

6.4.3 Strategic choice of PBs

During a simulation run, PBs make strategic and tactical choices. Strategic choices involve a reconsideration of their Business Model and allow PBs to adopt a different MM, product-based SM, service-based SM and level of production capacity. Strategic changes are always based on a trade-off between switching costs, expected sales and strategic 'fit':

- Switching between MMs and SMs involves *switching costs* related, for instance, to physical changes of the production location and equipment and to (re)negotiations of contracts with suppliers. Switching costs may also involve the PB obtaining required Skills and possibly obtaining access to certain required Infrastructures.
- PBs can estimate *expected sales* by performing market research among a certain fraction of market participants, chosen such that part of them are current customers. The results from the market research provide the PB with estimates on expected sales at each price level, allowing the PB to optimise expected profit.
- In order to determine a *strategic fit*, each SM has a set of 'scores' on the same set of properties that define a PB's strategic beliefs. The scores indicate how well the SM addresses each specific preference aspect (e.g. status level, environmental friendliness). PBs will favour SMs that score well on the aspects they find important, even if the expected short-term profit is somewhat lower than that of SMs with a lower fit. This leads to heterogeneity in choices that can also be observed in reality.

6.4.4 Tactical choice of PBs

More frequently than the strategic reconsideration, PBs perform tactical reconsiderations where they set selling price and target stock levels such that they can expect to make maximal profit given their current Business Model configuration of MM, SM(s) and capacity level. The price optimisation is again based on market research among a fraction of the market participants. The PB then sets its target stock levels somewhat higher than strictly necessary for accommodating expected sales, to account for the present uncertainty. The tactical reconsideration process allows PBs to stay competitive and change prices when forced to by their competition.

6.4.5 Innovation

PBs do not have an R&D department in the model. Instead, innovation is represented as an external Market Development event that may occur at a given time or when a certain condition is met (e.g., if some Product

establishes a prespecified market penetration level). Such an 'innovation' event makes some previously unavailable Product or Service, as well as the related MM, SM and CM, available as a new option that businesses and consumers can choose to sell or consume. Again, the Product or Service and all associated models must be predefined by the user: innovation events just switches their visibility on or off for agents in the model. The model cannot dynamically generate business models that were not explicitly defined as input data.

6.5 Consuming agents

6.5.1 General characterisation

The model has two types of Consuming Agents (CAs): Consuming Businesses (CBs) and Consuming Individuals (CIs). All CAs need to satisfy, at every time unit, a certain amount of the same central Need. They have a Consumption Model (CM) that transforms either a certain amount of a Product (quantity or use time) or a Service, both obtained from PBs on the market, into the Need. Similar to what the previous section described for PBs, preferences of all CAs relate to certain quality aspects of products and services, but may also include environmental aspects, market penetration (in this case representing social conformity) and degree of loyalty to current suppliers.

6.5.2 CIs and CBs

CIs are characterised by their lifestyle, which defines their required amount of Need per time step, their available budget for fulfilling the Need, their preferences with regard to CMs and their willingness to pay for offers that provide a better fit with those preferences.

Although CBs have strategic preferences instead of a lifestyle, these preferences correspond directly with the lifestyle aspects of CIs. For instance, a Consuming Business can also have a preference for consumption models that score well on environmental aspects. The only effective difference between the two types of agents in the model is that CIs try to minimise costs while CBs try to maximise profit. This is different because CMs may also affect the profit of CBs by modifying their output capacity. For instance, in the agri-food case, one PB could represent a grower of tomatoes. The PB has to choose between do-it-yourself (DIY) pest protection or an integrated pest management (IPM) service. IPM, although possibly more expensive, is not unlikely to lead to higher tomato sales compared to DIY, as the specialised care offered by IPM may lead to a higher yield of higher-quality tomatoes. The SPREE model allows CBs to make such a trade-off between additional costs and additional sales.

6.5.3 Strategic choice of CAs

Periodically, and whenever a durable product has reached end-of-life, a CA will engage in a strategic re-consideration, re-evaluating all CMs for which there are offers on the market. In the current version of the model, CAs still have full knowledge of all these offers and objectively discern their properties. When choosing between CMs, the CA first filters available CMs on budget violations and preference threshold violations. If multiple options qualify, the agent compares the remaining CMs in a pairwise manner. For each pair, the CA makes a trade-off between the costs of both offers, the normalised sum of their weighted scores on preference aspects, and a possible correction factor for loyalty.

6.5.4 Tactical choice of CAs

More frequently than going through the elaborate strategic reconsideration process discussed above, CAs may check if any other PB offers the required input Product or Service for their current CM at a lower cost. If the cost difference exceeds the consumer's loyalty factor, she will switch to the cheaper supplier. This tactical reconsideration is not directly relevant for owned durable Products, but Services typically have much lower switching barriers and are paid for per time step. The loyalty factor is an important stabilising factor, as it prevents CAs from constantly switching between suppliers.

6.6 Important limitations

6.6.1 Comprehensive, but still limited

At the moment of writing, the SPREE generic model is probably a more comprehensive agent-based integration of concepts related to the behaviour of participants in real markets than any of the models found in literature (see chapter 5). The only other very extensive model of markets is the EURACE model (see 5.6.5), but that model focuses on large-scale interactions based on GIS data and is much less subtle in its agents' decision making processes. Moreover, it actually consists of a *set* of coupled models while the SPREE model integrates all aspects in just one model.

Despite all the concepts that the SPREE model does somehow account for, it has important limitations. These limitations relate to scope, network and geographic aspects, and excluded concepts and mechanisms. The concepts below were not excluded because they were irrelevant, but mainly because of practical considerations. Other concepts were simply considered more important to include in the model.

6.6.2 Scope

The SPREE generic model allows for interactions between a fixed number of Businesses and Consumers on one virtual marketplace. It allows to study the effects of Policy Instruments under varying market development scenarios. The model does not currently include:

- *Market entry and leave.* Even if Businesses make large amounts of losses, or if Consumers cannot satisfy their Need within their budget, they will never leave the market. New agents cannot currently enter the market either.
- *Supply chain interaction.* The marketplace represents one link of a supply chain. Everything that happens outside this link (i.e., upstream and downstream in the chain) is represented by the World Market.
- *C2C markets.* Servicising in practice can also occur when individual Consumers mutually exchange Products and Services. The model does not include such interactions.
- *The legislatory and social implementation process* that precedes the effectuation of policy. This - also very important - part of policy development is the domain of other SPREE partners. The model only serves to simulate *effects* of implemented policies.

6.6.3 Network and geographic aspects

The SPREE generic model can be fully paramatrised based on the applicable conditions in a certain geographic locations and sector. Within that area and sector, agents:

- Have no *geographically explicit location* that in any way affects which agents communicate with each other, and to what agents certain regional policies may apply. Such details can be partially included by a clever parametrisation, but the model does not specifically support it or show it in the outcomes.
- Are not connected through any *explicit network structures*, such as small-world networks or networks of agents with a shared lifestyle or strategy.

6.6.4 Excluded concepts

Looking back at the wide range of concepts addressed in previous studies (see chapter 5), the SPREE generic model unavoidably misses out on some concepts that may also affect servicising shifts:

- *Bounded consumer rationality.* Consumers in the SPREE model have full knowledge of all offers on the market and their properties.
- *Subjective perceptions.* Agents in the SPREE model all perceive product properties in the same way, although they may weigh the properties in different ways when making decisions.
- *Learning.* Agents in the SPREE model do not look back to evaluate historic success. Instead, they look ahead, by means of cost calculations and market research. This was a deliberate modelling choice, based on the observation that forecasting is very important in many markets, while existing models seldom address this aspect.
- *Detailed contracts and risk.* Discussions with SPREE participants indicated that contract specifications and risk sharing are important aspects of business models. However, it turned out that the theoretical basis to integrate these concepts with the whole set of other concepts (especially agents' decision making logic) was lacking. Also, required data for such considerations would be impossible to obtain within the scope of the SPREE project. Contracts in the SPREE model are therefore relatively simplistic representations of reality, and risk (sharing) is not explicitly addressed at all.
- *Finance and employment.* For the same reasons, finance and employment aspects have also been completely excluded from the model.

6.7 Conclusion: the model in a nutshell

This chapter has provided a high-level overview of the SPREE generic model. The model is descriptive in nature, keeping as close to real systems as possible while still being generically applicable to different markets. The model already includes much of the functionality that may be required in the three case studies.

The model represents a generic market of sellers and buyers that revolves around some central consumption Need. Both the sellers and buyers make strategic and tactical level choices. Their decisions are based on trade-offs between monetary aspects and personal, social and/or strategic preferences.

Figure 6.5 summarises all direct relations between agents and objects, as well as between objects an other objects, in one diagram. PBs choose an MM and up to two SMs to be able to sell Products and Services to consuming agents (CAs), which in turn have a CM to satisfy their Need. Agents can connect to Infrastructures and learn Skills that both function as prerequisites for CMs, MMs and SMs. Products and Services originate from the World Market, and can be either sold to the World Market or dumped in the Physical Environment. Products and Services have associated material extraction and generated waste in

the form of a set of Resources. Service Contracts are links that represent an agreement between a PB and a CA on the delivery of a Service over a longer time span (for esthetic reasons, the figure does not explicitly relate Service Contracts to CAs and PBs). Finally, the Observer controls Policy Instruments and Market Development Events, the Effects of which can influence any property of agents and objects in the model.

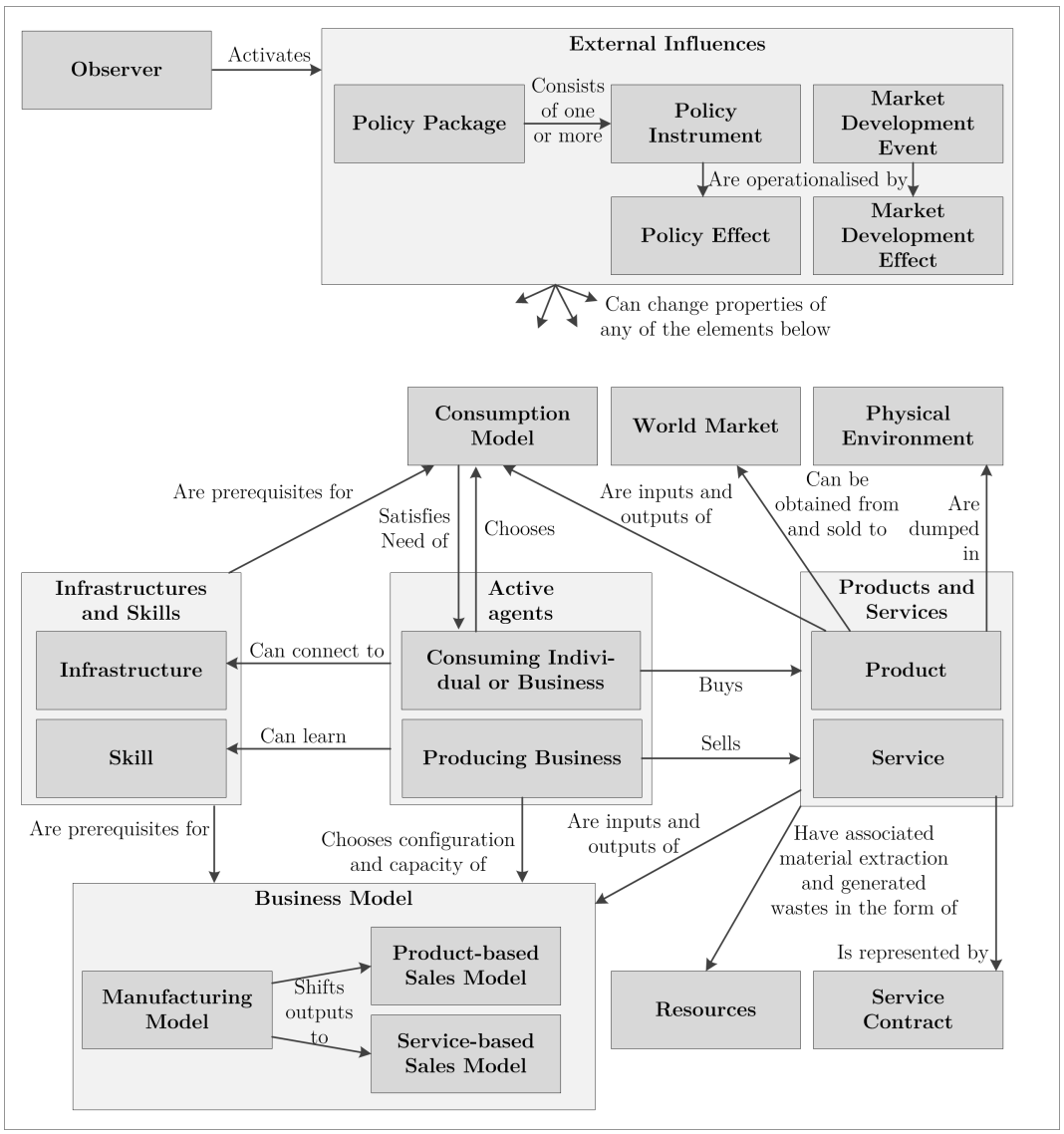

Figure 6.5: All model elements in one diagram

Despite its integration of relatively many concepts, the model still has important limitations with regard to scope, network aspects and geographic aspects. Some other possibly relevant concepts, mainly relating to subjectivity aspects and risk and finance aspects, had to be excluded as well.

7 NetLogo implementation

7.1 Introduction

This chapter discusses details about the software implementation of the conceptual model presented in chapter 6. Section 7.2 first discusses the general approach used during the implementation process. Section 7.3 briefly discusses NetLogo, the implementation platform used for the SPREE generic model. Section 7.4 discusses the meta-structure of model objects, types and instances. Section 7.5 elaborates on technical details such as the size of the model, naming conventions, code documentation and the use of supportive variables. Section 7.6 discusses the initial state of all model elements. Section 7.7 presents the model interface, with the model dashboard visualising what happens during a simulation run by displaying the agents and their choices, as well as several charts of aggregate statistics. Section 7.8 discusses the verification checks that were performed to test whether the model has been built right. Finally, section 7.9 discusses planned future refinements.

7.2 Implementation roadmap

7.2.1 Deep complexity

When looking at the number of model elements provided in appendix B, at the rich specification of each element, and at the variety of concepts addressed, it becomes clear that the SPREE generic model is a very extensive model, especially when comparing to the models reviewed in chapter 5. One does not simply start building a model of such proportions. The modeller needs a structured approach or he/she will get lost in all the interrelations and model complexity. The more complex the model, the more likely it becomes that the modeller is unable to write flawless code, and the more difficult it is to trace the origin of introduced 'bugs', i.e., faulty model behaviour. Complex models practically demand that bugs are identified as soon as possible after the faulty code has been written. This means that all code should be tested and debugged

shortly after it has been written.

If the system consists of interrelating subsystems each with a clear scope, the modeller can create different submodels for each subsystem, test these models separately and connect them when they are all verified as much as possible. However, the SPREE generic model consists of just one indivisible system of interactions. The complexity can be found mainly in the *depth* of the procedures themselves. A good example is the strategic reconsideration procedure (level 1) of PBs, that depends, among other things, on a comparison subprocedure (level 2) which in turn requires the calculation of profits (level 3) based on the creation of a demand curve (level 4) from market research (level 5). During the market research, consumers are asked at what price they would switch to a certain product. They arrive at this price by using a version of the CA comparison procedure (level 6) that depends, among other things, on the calculation of total expected costs over the reconsideration period (level 7). This example demonstrates the deep complexity present in the SPREE model.

7.2.2 The SPREE model implementation approach

A very naive approach would be to simply build all the procedures according to the model formalisation (i.e., the narrative, flowchart, pseudo-code), without any intermediate testing. When done, the modeller would press 'go' and probably finds out that the strategic reconsideration procedure (or some other high-level procedure) encounters an error because some value is outside some acceptable range. Since model information propagates through all subprocedures, the error can now be located in *any* of the seven levels! Especially when considering that the subprocedures actually constitute pyramid structures, with multiple lower level procedures on each level, tracing the origin of the error can take days or even weeks. And many more bugs will be discovered than just one.

The SPREE generic model implementation therefore closely followed a very explicit 'implementation roadmap', depicted in figure 7.1. The roadmap divides the implementation process into seven phases (not corresponding to the seven levels discussed above), starting at the lowest level procedures and then enriching the model in each new phase by adding more complexity. At the end of each phase, the model had to be fully functional before continuing to the next phase. This means that a test round took place between every two phases. Newly identified bugs during those test rounds were always the result of (interactions with) the newly added elements, relations and procedures.

Implementation roadmap	
1. Static market interaction	• MM, SM (only 1), CM all fixed • Consumers and Consuming Businesses • Buying and selling inputs and main output • Keeping track of money flows
2. Add CM reconsiderations and preferences	• CM no longer fixed: Consumers reconsider consumption strategy • Add CA Skills and Infrastructures • Add Consumer preferences to calculate lifestyle fit score • Add CB preferences to calculate strategic fit score, add threshold checks, add cloning, add CB profit calculations
3. Add reconsideration of price and production volume	• Allow PB to optimize price and production volume • Add market research • Add PB profit calculations • Add 'best offer' logic for Consumers
4. Add partial servicising	• Upgrade everything above to allow partial servicizing
5. Add strategic reconsideration for PBs	• Add MM properties per capacity • Add PB Skills • Add capacity, SM and MM investigation procedures
6. Add policy and market developments	• Add (de)activation logic for market developments and policy packages/instruments • Add applicable-policy-effects logic to Consumers • Add propagation chains of availability
7. Detailed analytics	• LCA • Product disposal

Figure 7.1: Implementation roadmap

7.3 NetLogo

7.3.1 Implementation platforms

The book by Nikolic et al. introduced in chapter 1 proposes three possible ABMS implementation 'platforms': NetLogo, Repast and custom code. Many more modelling environments are available. NetLogo is one of the most user-friendly environments, but this comes at the price of a lower degree of programming freedom and limited scalability. It has a low entry barrier for new modellers but is also used by more experienced modellers, mainly for proof-of-concept models and rapid prototyping, or during a demonstration of ABMS.

Repast offers the user more flexibility and control over the agents in the simulation, but is less accessible and requires a working knowledge of Java to use effectively. The third option, custom code in a high-level language such as Java or C++, offers even more flexibility and even less support. This option may be necessary to accommodate integration of the model with other software applications and/or software

libraries.

7.3.2 Choice for NetLogo in SPREE

For the SPREE project, we have have selected NetLogo as the implementation platform, for the following reasons:

- NetLogo is open-source (Repast and Java environments are also freely available, but we also considered some paid platforms).
- I already had experience with NetLogo, reducing the learning curve involved with programming
- NetLogo features a fairly simple programming syntax with keywords that specifically support agent-based control and interaction. This makes the model easier to communicate on a detailed level.
- We were curious if NetLogo would also be suitable for building larger models than just prototype, proof-of-concept models.

According to the book by Nikolic et al., NetLogo's relative simplicity puts severe limits on the complexity of models that can be created. However, in our experience during the implementation of the SPREE generic model, this has not been a problem. Still, the specific data structures and programming practices in NetLogo may limit the maximum feasible model size in terms of number of agents (more so than e.g. Repast).

7.4 Meta-structure

When making the step from model formalisation to implementation, the modeller has some freedom with regard to the meta-structure of the model to be implemented. Meta-structure here refers to the high-level structure of storing into one model structure the different agents, objects, environment, but also shared properties of agents and information tables. Figure 7.2 shows that the SPREE generic model consists of four basic model elements: the observer, agents, objects and links. There is only one observer, which can influence the model (e.g. through the activation of Policy Instruments) but does not interact with agents in the model. For agents and objects, several *types* exist, such as 'Producing Business' or 'Consumption Model'. These types correspond to the model elements identified in chapter 6. The type determines the set of attributes shared by all library agents/objects and instances of that type. The only actual 'links' in the model are the Service Contracts.

Only *instances* of agents and objects can directly interact in the model. Library agents/objects just contain information. They specify shared attribute *values and value distributions* for subgroups/subtypes

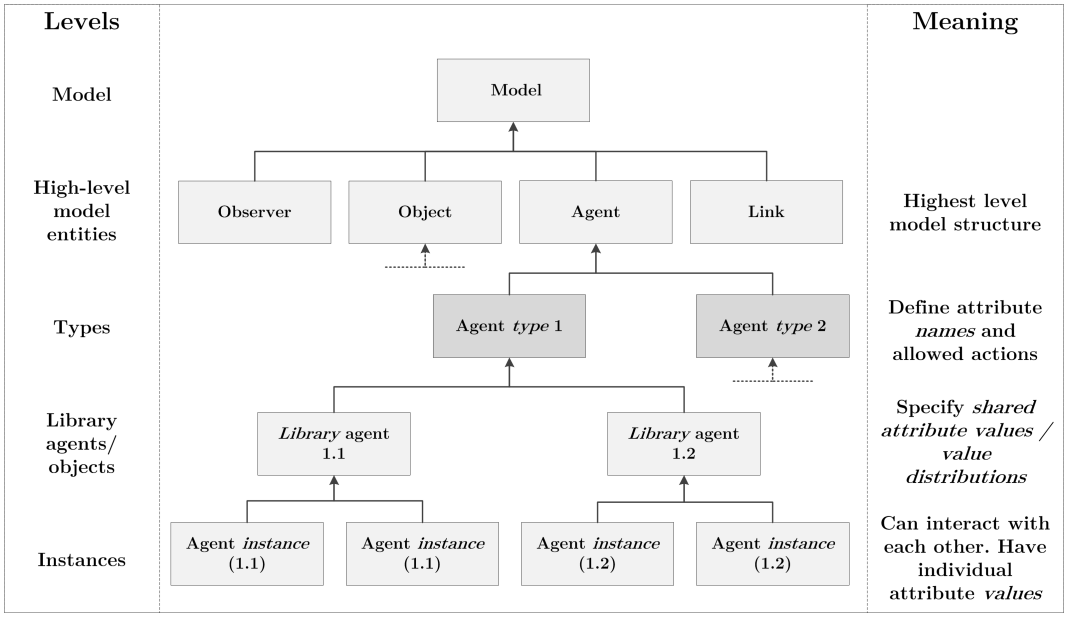

Figure 7.2: SPREE generic model meta-structure

within a type. For example, there can be a library agent for a user-defined group of 'environmentally aware' PBs. The library agent contains the attribute values shared by all *instances*, i.e. individual representatives of that group in the model. Some values may be specified as stochastic distributions to allow for heterogeneity *within* groups. An example of a library *object* is a Product subtype 'Small electric vehicle', containing the information shared by all instances of this Product subtype in the model.

Every NetLogo model consists of an observer, turtles, links and patches. The observer corresponds to the Observer identified in chapter 6. Each type of agent or object is represented as a different 'breed' of turtles. Library agents/objects are modelled as distinct breeds, as they may contain distributions instead of sampled values. The only type of links in the model are Service Contracts. However, additional one-directional relations are often established by adding an element instance as a state of another agent. For example, a PB has a state 'a_Manufacturing_Model' (with 'a_' indicating that this is an attribute) that contains a direct pointer to an MM instance in the model. This way, the model 'knows' that the PB and the MM are related. The fourth high-level type of NetLogo elements, i.e. the 'patch', is not used in the SPREE generic model at all and therefore not relevant to discuss.

7.5 Technical details

7.5.1 Model proportions

Appendix E contains the full code of the SPREE generic model as implemented in NetLogo. The appendix also contains a reading guide to help understanding the NetLogo programming language and the naming conventions used. The model code consists of 5241 lines, of which 1057 are empty line breaks, 727 only contain documentation and 3457 lines consist of actual code (sometimes also containing clarifying documentation), divided over 178 procedures (including some technical support procedures and plotting procedures). Although this gives an impression of the proportions of the model, it is not a perfectly objective measure for comparisons with models built in alternative environments such as Java. The modeller always has a lot of freedom in the degree of programming compactness and efficiency. In addition, NetLogo is much more *expressive* than higher-level languages such as Java. This means that the modeller can give the same commands within fewer lines. This property derives, for instance, from NetLogo being much less demanding with regard to definitions and declarations of procedures, elements and variables. NetLogo also allows for much more compact if-else statements and loops.

7.5.2 Content-related remarks

Two other technical implementation aspects are worth mentioning here:

- Agents take actions in a random order that is different in each time step. This is required to prevent systematic first-mover advantage of certain agents, which could, for example, lead to a certain group of agents that always has last choice among the available stocks on the market. Such results might be wrongly interpreted, but as said, the random order of action prevents it from happening.
- During market research, the interviewed CA must know the best alternative offer on the market to compare to. Instead of always performing a full comparison of all offers on the market, CAs simply keep track (through a dedicated property/state) of the best offer for them at the market at any time. This offer can be completely different from their current CM, as CAs do not directly switch to a new offer at the moment it becomes available, but only periodically (see conditions for CA strategic reconsideration in section C.5.1). Whenever a new offer enters the market, all CAs compare it to their current best offer according to the comparison procedures described in boxes C.9 and C.10.

7.6 Model initialisation

The behaviour of agent-based models can be very dependent on their initial state. It is therefore important to understand how the model initially populates with agents and objects and how their properties are set.

7.6.1 The SPREE data input file

The model depends on a big structured input file with flat text in a fixed format to allow the model to correctly read the input data. Every parameter that is relevant input for a specific case study has to be defined here, because model variables (e.g. globals, states, properties) have no 'default' values other than 0. For each type of agent or object, various subgroups can be defined as a new set of parametrisations of the relevant states for that type. Some input fields accept input in the form of minimum and maximum values, thereby specifying the range of heterogeneity between agents within a subgroup. Box 7.3 provides an example of a fully parametrised input group. The example is from the fictive 'Dutch cycling' case study, introduced in chapter 8.

7.6.2 Creating library agents/objects and instances

Without going into the plausibility of the provided numbers in box 7.3 (which are fictive anyway), the example makes very clear how much input data the model needs, and how the required amount of data explodes when more groups and preference aspects are introduced. The example also shows that some parameters are assumed exactly the same for all agents in the 'cheap and quick' group (e.g. 0.05 as the relative willingness to pay for loyalty), whereas other parameters are defined as a range of possible values (e.g. 4-5 for the weight of the first preference aspect). The model interprets this range as a uniform distribution, i.e. equal probability for each value in the range.

The model initialisation primarily constitutes of processing the entire input file. For each new group, the model first creates a library agent/object and then creates the specified number of instances for that group (50 in the example). Every instance (i.e. each individual agent or object) is parametrised based on the respective values from the input data, sampling from a random uniform distribution where applicable. For PBs, the input file precisely specifies the initial business models and selling prices for each agent instance.

7.6.3 CA initialisation

The initial CM of the CAs is not predefined. Instead, once the model has created all PB instance, CAs autonomously select the most attractive offer on the market and adopt the corresponding CM. This eliminates

Consuming Individual group parametrisation example

Consumer Group ID
1

Name
Cheap and quick

Initial number of Consumers in Group
50

Initial percentage of Consumers that could be connected to each of the relevant infrastructures [Infrastructure ID, fraction]
[[1 1] [2 1]]

Initial percentage of Cs with infrastructure availability that are actually already connected [Infrastructure ID, fraction]
[[1 1] [2 0.95]]

Initial Skill IDs [Skill ID, fraction]
[[3 1] [4 1]]

Need (f.u./BTU) [minimum, maximum]
[1 2]

Maximum threshold for total costs per functional unit of need (euro/functional unit of need)
5

Willingness to pay for keeping with the same supplier (loyalty), relative to the overall price per btu
0.05

Willingness to pay for 1 point higher lifestyle fit, relative to the total consumption cost per btu
0

Weights for each of the preferences and (last value) market penetration
[[4 5] [0 1] [1 2] [4 5] [0 1] [0 1] [3 4] [0 1]]

Minimum thresholds for each of the preferences and (last value) market penetration
[[0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [0 0]]

Return-on-investment period (BTU)
520

Period after which Consumer reconsiders a strategic change (BTU)
104

Period after which Consumer reconsiders supplier (BTU)
52

Box 7.3: Example parametrisation of a CI group

the need to specify initial CMs of CAs and also provides a validation check for the user: if the initial choices are close to the choices that market participants make in reality, this increases the user's confidence in the plausibility of input parameters. For all product-based CMs, an initial associated product is created with a randomly generated remaining use time. For all service-based CMs, the model creates the associated Service Contract.

Next, the PBs find out how many CAs have selected their offer as best. The PBs then calculate appropriate capacity levels and target stock levels. The stock targets are assumed to be fully met at the start of the model simulation, so the model creates a number of corresponding Products. The initial remaining times

until tactical and strategic reconsideration are also randomised for all agents. Policy Packages/Instruments and Market Development Events are not assumed to be active yet. They may, however, activate during the first time step.

7.7 Model interface

7.7.1 Visualisation of model elements

Figure 7.4 shows a screenshot of the model dashboard, populated based on the 'Dutch cycling' case study described in the next chapter. The red numbers have been inserted later to identify the different areas of the dashboard.

Figure 7.4: The dashboard of the NetLogo implementation of the SPREE generic model

The figure identifies six separate areas (requires viewing in full colour to see properly):

1. Control area. The user can select an experiment here, initialise the model by pressing 'Setup' and then let it run step by step ('Go') or for 2000 time steps at once ('Go-2000'). This takes approximately 1 minute in the 'Dutch cycling' case, with 5 PBs and 126 CAs, on a third generation Intel Core I7 processor.

2. Policy area. The user can currently manually activate policies and determine their start and end times, but this will eventually be controlled through the input files. The red squares turn green while the respective policy is active.

3. Market developments. This area shows a list of the most recent market developments and their time stamp (but is currently empty in the screenshot).

4. Sellers. This area shows the PBs as big grey boxes, grouped according to the defined PB groups, with lighter shades of grey if the PB engages in partial or full servicising. The arrows within the boxes represent the agent's MM and SM(s). The small boxes on both sides of (some of) the arrows represent stocks of Products, with colours corresponding to the colours in area 6.

5. Buyers. This area shows the CAs, again as grey boxes grouped according to the defined CA groups. Light grey indicates that the agent‘s CM is service-based. The arrows here represent CMs, with colours corresponding to the corresponding SM of the agent‘s current supplier. If the agent‘s CM is product-based, a small box in the upper left corner of a CA represents its currently owned Product. The colour of the Product corresponds to the colours in area 6. In case of a service-based CM, a box in the upper-right corner represents the corresponding Service Contract, again with colours that correspond to area 6.

6. Market shares. This area shows the available set of Products and Services in the model, and the current market share of each, based on the sum of the total Need satisfied at each time unit by CMs depending on that Product or Service.

The NetLogo environment allows for a more detailed inspection of the elements in areas 4 and 5, with a pop-up screen allowing the user to inspect all of its states/properties at any time during the simulation.

7.7.2 Charts

In addition to the model dashboard, the model interface features a number of charts that generate while the model is running. These charts directly address identified concepts and desired functionality from chapter 4. Table 7.1 lists the charts currently present in the implemented model, and specifies briefly how they relate to concepts and functionality. The next chapter contains examples of what the charts may actually look like, based on the illustrative case.

The charts can be tweaked by users with NetLogo experience if needed for a specific case study. Additional charts based on existing agent properties can be created with relative ease, and currently defined charts can be removed even more easily. The variety of possible charts illustrates for many aspects of desired functionality identified in chapter 4 how they are addressed by the model.

Charts in the SPREE generic model		
Name/description	**Related concepts**	**Related functionality**
Price developments of individual Products and Services, and overall per type.	Competition Consumer spending	Competition between products and services
Market shares of individual SMs, Product types and Service types	Competition	Competition between products and services
PB choices	Business model	Explicitly address servicising
Overall fraction of consumption that is servicised	Ownership-oriented vs function-oriented	Explicitly address servicising
Total profit of each PB	Business profits	Insight in economic effects
Average money spent per time step by CAs	Consumer spending	Insight in economic effects
Average time spent by CAs	Quality of life	Insight in social effects
Average lifestyle fit score of CAs	Quality of life	Insight in social effects
Aggregate resource use per time step	Resource use	Insight in environmental effects
Aggregate amount of waste per time step	Generated waste	Insight in environmental effects

Table 7.1: List of charts in the NetLogo implementation of the SPREE generic model, and how they related to identified concepts and functionality

7.8 Model verification

7.8.1 Verification components

This section aims to build confidence in the model by carrying out a number of verification tests. Verification is meant to confirm that the implemented code corresponds to the specifications defined during the model formalisation phase. These specifications are provided in chapter 6, supported by appendices B and C. The book by Nikolic et al. suggests several verification tests:

1. Single-agent testing, in which the behaviour of a single agent is verified.
2. Recording and tracking agent behaviour, in which relevant metrics are identified and recorded.
3. Interaction testing limited to a minimal model, in which the interaction between agents is tested.
4. Multi-agent-testing, in which the emergent behaviour is examined.

This model verification section addresses the first three items in the list. The fourth item is addressed in the next chapter, which examines model behaviour based on an illustrative case. Model verification activities were actually conducted throughout the development of the SPREE model, not just at the end. As described in section 7.2.2, the model was built in seven phases. At the end of each phase, we (the modellers) examined if all new functionality worked correctly, using the four techniques listed above.

It is important to stress that model verification is never really complete, as there are infinite combinations of input parameter values that may cause unexpected and undesired model behaviour. Still, every additional test further increases the confidence that the model has been built correctly. A certain level of verification is absolutely necessary before the model should be handed over to the user, although it is impossible to tell how much is 'enough' (Warnier (2013), personal communication).

7.8.2 Single-agent testing

This first type of verification tests is meant to verify the behaviour of low-level procedures and can be performed without further model context. It tests for a range of possible inputs (preferably including extreme values) whether the procedure calculates the correct output values. The verification is based on a comparison with calculations by hand or in MS Excel. Table 7.2 lists the procedures that we have tested this way. Each of these procedures is a critical building block of higher level procedures in the model. Results of the tests are available on request.

Isolated unit tests				
Procedure to test	**Executed by**	**Inputs**	**Outputs**	**Tested OK?**
calculate-total-net-switching-costs	PB/CB/CI	Old MM/SM/CM New MM/SM/CM	Total switching cost	V
calculate-max-preference-threshold-violation	PB/CB/CI	MM/SM/CM	Max. pref. threshold violation	V
calculate-preference-fit	PB/CB/CI	MM/SM/CM	Preference fit	V
calculate-variable-cost-per-unit-SM	PB	SM	Variable costs per SM output unit, incl MM conversion	V
update-target-stock-size	PB	SM, expected sales volume per time unit	Target stock size	V
calculate-total-required-new-n-products-in-period	CB/CI	CM, consideration period	Number of products to be bought during the period	V

Table 7.2: Isolated unit testing verification experiments for the SPREE generic model

7.8.3 Recording and tracking agent behaviour

Another important verification test is to log and inspect the states (or attributes) of individual agents and objects over time, as they participate in model procedures. This type of test was standard practice during the implementation process of the SPREE generic model and was the main method of tracking down bugs. We routinely created output lines at multiple points in a new procedure. We then ran the model, examined the output logs and again compared them to calculations by hand (or in e.g. MS Excel). Also, whenever we observed bugs causing unexpected model behaviour, the inspection of logs was our main method of tracking down these bugs.

This verification technique was especially essential for the verification of model procedures relating to market research and price optimisation. It was used to verify, for example, the following model procedures:

- Generate a list of possible business model configurations (PB) - given the current configuration
- Compare two business model configurations (PB) - given that the maximal profits over the ROI period, the maximum preference threshold violation, the preference fit, and the loyalty discount have already been calculated
- Calculate the price that maximises profit for a business model configuration (PB) - given all information gathered during market research
- Production processes (PB)
- Filter CMs based on infrastructure availability (CA)
- Calculate total costs per time unit for a given CM (CA)
- Compare two offers (CA) - given that the total costs (or profits) per btu, the maximum preference threshold violation, the preference fit, and the loyalty discount have already been calculated
- Calculate the switching price when asked during market research - given a current best offer, a new offer, and relevant properties of both (CA)
- Calculate market share scores - given the choices of consumers
- Apply effects of policy activation

7.8.4 Interaction testing

The aim of interaction testing is to verify the interaction between agents. In the SPREE generic model, interaction primarily occurs between PBs and CAs. CAs choose between offers of PBs, and PBs perform market research by 'interviewing' CAs. The two associated model procedures are the most complex pieces of code in the SPREE generic model. They are difficult to verify, because they rely on many layers of subprocedures and have to process relatively large amounts of information.

We have tested these two procedures in the same way as the procedures in section 7.8.3, i.e. based on output logs that describe successive states and actions of agents. The resulting logs are just more elaborate, but the principle remains the same.

7.9 Future refinements

Although the current implementation of the model is relatively complete and working, a number of refinements is planned for the short term and should be completed at least before doing the SPREE case studies. The refinements include:

7.9.1 Policy-related refinements

The following refinements will make the parametrisation of policy more flexible and user-friendly:

- The 'progress scenario' procedure that updates activation of Policy Instruments and Market Development Events is currently not implemented as described in section C.3. Instead, the Policy Instruments for the example 'Dutch cycling' case are hard-coded in the model. Completing the implementation as envisioned before has high priority. A challenge here is how to allow, in a general way, the reversal of temporary Policy Effects. However, the design plans for that specific task have already been conceptualised and are ready for implementation.

- The model should have automatic propagation chains of availability states. If a Policy Instrument bans a certain Product, the model currently does not understand that this also makes the associated SMs and CMs unavailable. Whenever the availability of some object changes, the model should automatically check for any elements that depend on that object and update their availability status if needed.

7.9.2 Refinements relating to the central comparison procedure

The three refinements below provide (possibly superior) alternatives for the central comparison procedure of both producing and consuming agents:

- Preference thresholds are currently calculated as *relative* thresholds. As a result, a score of 1 on a threshold of 2 counts as a 50% violation, whereas a score of 5 on a threshold of 8 counts as 'just' a 37.5% violation (while the threshold value of 8 is very high, indicating that the preference aspect must be important to the agent). It is probably better to consider *absolute* preference threshold violation, such that the violations are 1 and 3 in the examples, respectively.

- Also relating to thresholds, agents currently just consider the single highest violation. This means that if there are multiple violations, only the highest (relative) violation counts. An approach that sums the squared threshold violations on all aspects is probably closer to reality. It does not seem unlikely that consumers take all 'violations' into account in their decision-making, while assigning a higher than proportional weight to large violations (hence the squared values).

- The pairwise comparison procedure that structures strategic reconsiderations has a practical problem (which fortunately does not occur too often). The problem is that the comparison of three different options is not necessarily transitive. This means that in a full pairwise comparison, option A may be better than option B, option B better than C, but option C better than A! We know the cause for this problem, but solving it will require structural changes to the comparison procedures, while the amount of additional stochasticity that this 'problem' introduces is completely overshadowed by all other stochasticity in the model. One may even consider it as a good feature representing actual consumer non-rationality. In reality, consumers may also compare three options in a pairwise manner, based on non-transitive criteria. The order of comparison, instead of some 'objective' overall score, may then determine the outcome.

7.9.3 Refinements relating to business models and PB reconsideration

The following refinements will make the model conceptualisation of business models more realistic:

- Services can currently be delivered by PBs as long as there is sufficient stock of the required input Product (if any). However, in reality the required 'stock' level may be much higher. For example, if 10 people rent a car on a given day, the number of needed cars is probably 10 (or somewhat less if they are only rented for a couple of hours). In contrast, the model assumes that one car can do the

job, as long as it has sufficient use time. This is a limitation of the current implementation that can be mitigated by introducing, for instance, a 'stock to demand ratio' that determines how much stock the PB needs for each unit of Service delivered during one time step.

- The PB currently has a risk factor that affects estimated sales equally for all considered SMs. It may be more realistic to apply a lower risk factor to calculated sales for the SM that the PB currently has. Alternatively, the PBs could take into account historic sales of that model as an indicator of future sales.

7.9.4 Strictly technical refinements

Finally, the two technical refinements below will increase scalability of the model:

- Although the model procedures already have relatively little overhead, the model probably still has much potential for speedup. The three case studies may well involve more agents than the 5 PBs and 126 CAs present in the illustrative case study used for model testing and verification (see next chapter). Any successful speedup effort will allow to do more experiments and/or larger study populations within the available time, and may therefore be well worth looking into.
- The visualisation settings currently only allow for a maximum of approximately 50 PBs and 200 CAs. In order to facilitate larger-scale models, the visualisation should automatically adjust based on the number of agents to be displayed.

7.10 Conclusion

This chapter has discussed details of the implementation of the SPREE generic model. It described the roadmap used to structure the implementation process in a suitable way to tackle the deep complexity in the model. It introduced NetLogo, which was selected even though it may have some practical limitations, and presented the meta-structure of the model objects, types and instances. Furthermore, the chapter has dealt with technical implementation details such as model proportions and programming practices. It described how the initial state is almost entirely dependent on input data, but that CAs make some initial choices that directly provide a validation check on the input assumptions. The chapter then presented the model dashboard of the implemented model and described a number of charts that it can generate. The section on verification discussed three types of verification tests that have been conducted on model procedures, with different approaches for simple and for complex procedures. The final section discussed planned refinements of a varying nature.

The following chapter discusses the model behaviour based on the fictive 'Dutch cycling' case. It thereby provides an elaborate fourth verification step and illustrates model functionality. It then evaluates how much of the desired model functionality has actually been realised.

8 Model behaviour

8.1 Introduction

With the model implemented as described in chapter 7, this chapter will explore the model behaviour. It defines a fictive but complete 'Dutch Cycling' case study to explore model dynamics and illustrate the types of analysis that it supports. Since all data are ficitive, the findings in this section are not suitable to derive actual conclusions on servicising dynamics and policy. Section 8.2 introduces the illustrative case. Section 8.3 explores the model behaviour during a simulation run. Section 8.4 explores the effects of fictive policies to see how the model responds to external influences. Section 8.5 makes some additional high-level observations on patterns in model behaviour. Section 8.6 evaluates to what extent the model achieves all desired functionality identified in chapter 4.

8.2 Illustrative case study: 'Dutch Cycling'

8.2.1 General description

It is impossible to generate model behaviour without any input data. We (the SPREE TU Delft team) have therefore developed a fictive case study, entering fictive values, loosely based on reality, into all data fields that the model requests. Throughout the whole model implementation process, this fictive Dutch Cycling case was used to test if the model worked correctly. Appendix F contains all input data that parametrises the Dutch Cycling case study.

The case regards a bicycle market that revolves around the central Need of 'hours of bicycle use'. Bicycle producers and sellers constitute the PBs. The CIs consist of cyclists. There is one CB that represents a bicycle freight transport company. We assume that the bicycle producers directly sell to the end user. The possible servicising shift is from owning a bike to participating in bike sharing schemes. The basic time unit (i.e., the duration of one time step) is a week. 2000 time steps therefore correspond with roughly 40 years.

PBs can choose between three MMs, with the following three outputs:

- Bicycle A, a 'typical' Dutch bicycle
- Bicycle B, which is more expensive to produce but has better scores on some CA preference aspects
- Bicycle A2, a more environmentally friendly version of bicycle A that is, however, not suitable for sharing services.

For bicycle A and bicycle B, there are two possible SMs: one product-based and one service-based. The product-based SMs simply represent selling the whole bicycle. The service-based SMs have as output 'sold hours of bicycle use time', instead of the bicycle itself. For bicycle A2, only a product-based SM is available.

8.2.2 Agent groups

The groups of agents distinguished in the case study are not based on empirical data. They are somewhat plausible characterisations of participants in a bicycle market, designed to see how the model deals with heterogeneity aspects. The case study distinguishes two types of PB groups:

- 'Big' PBs. This group consists of two companies that are currently the market leaders, produce on a relatively large scale, do not often reconsider their strategy and assign less value to strategic preferences than to calculated (short-term) profit. They are assumed to already engage in partial servicising, meaning that they start with two SMs each.
- 'Small' PBs. This group consists of three niche-players. They are smaller in size, are more agile in their decision-making and have a specific strategy, which they hope will bring long-term success in the bicycle market. They assign relatively high value to aspects such as environmental impact, status and innovativeness in their strategic decisions.

The case study has one CB group, with only one agent instance:

- The 'Bicycle freight transport company'. This agent does not bother with environmental considerations and is mostly interested in short-term profit.

Finally, there are three groups of CAs:

- 'Cheap and Quick'. This group of 50 CIs is very cost-oriented, with a tight budget, and assigns high value to a low overhead time when using a bicycle. They basically just want to have a bicycle in the garage. This group will be the most unlikely to adopt service-based offers.

- 'Green & Trendy'. This CI group of 25 agent instances puts high value on environmental aspects of offers, as well as status and innovativeness. They are prepared to spend more money on offers that meet their preferences.
- 'Convenient'. This last group of 50 CIs also wants low overhead time when using a bicycle. In addition, they place a higher weight on the 'user-friendliness & comfort' level of the offer and are willing to pay some additional money to meet these preferences.

8.3 Observed model behaviour

We have simulated several runs with the Dutch Cycling case, given the input parameters specified in appendix F. We first just observed the model behaviour itself, and then did some runs with policy activation to see how that works out.

8.3.1 Stochasticity from the start

The first very important observation is that each model simulation run is unique. This is a result of all the stochasticity involved especially in the exact characterisation of agents. It means that one simulation run is definitely not representative and cannot serve as the sole basis to draw conclusions on the effects of a certain policy. The stochasticity involved is already observable when looking at the variation in initial market shares in 10 separate simulations (all with the same input data), as shown by table 8.1.

Offer type	Initial market shares in 10 separate simulations									
A selling	43.0%	45.1%	42.0%	45.9%	42.5%	44.7%	42.7%	49.3%	42.1%	43.2%
A2 selling	8.3%	9.6%	12.2%	4.5%	11.2%	10.3%	5.2%	2.8%	11.2%	8.9%
B selling	29.0%	24.6%	26.4%	29.5%	25.5%	24.9%	31.1%	29.0%	24.7%	26.0%
A sharing	11.8%	11.6%	14.6%	15.4%	13.9%	11.9%	13.7%	9.5%	12.1%	14.8%
B sharing	7.9%	9.2%	4.8%	4.8%	6.9%	8.1%	7.3%	9.4%	9.9%	7.1%

Table 8.1: Initial market shares of the Products and Services in the Dutch Cycling case, for 10 separate simulations

Despite the stochasticity in results, the model outcomes show consistency: the market shares distribution over the Products and Services is very comparable between the 10 simulations. The main reason is that although there is heterogeneity within groups, the groups as a whole have preferences within relatively narrow ranges, making the initial choices of the groups as a whole quite predictable.

8.3.2 Dynamic behaviour over time

When letting the simulation run for 2000 time steps without any external influences (e.g. policy), one can observe from the model that PBs almost never switch to a new business model configuration (at least not to a different MM) but that CAs switch quite frequently as a result of price volatility. Figure 8.1 shows the business model configurations that PBs adopted over the course of one simulation run (2000 time steps). The figure shows that PB4 (purple) switched from selling Bicycle A2 to offering Bicycle A quickly after the start of the simulation. PB3 (yellow) quickly switched to offering B, initially just as a Service and later also as a Product, until switching back to offering Bicycle A somewhere halfway the simulation. The 'Big' PBs, i.e. PB1 and PB2, never made a switch in MM, consistent with their characterisation.

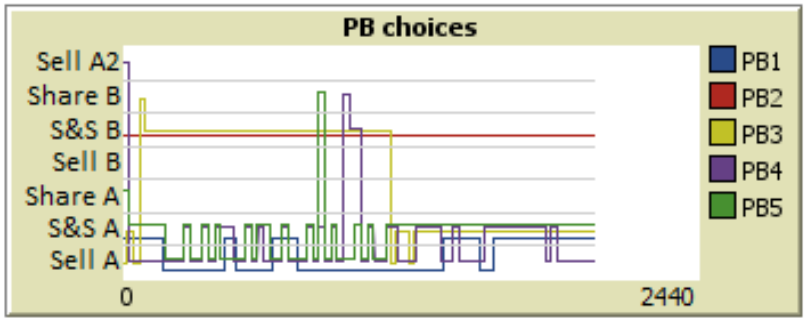

Figure 8.1: Choices of PBs over the course of the simulation. Each line represents the choices over time of one PB. The position of the line at each time step (x-axis) indicates the business model composition of the PB at that time step.

Figure 8.2 shows the price volatility for one of the runs. The upper chart, with prices of Products, shows the prices for Bicycle A (green/purple) and Bicycle B (red/yellow) - Bicycle A2 quickly disappeared from the market. The yellow line seems to 'stop' somewhere halfway but continues among the other lines at the bottom half of that chart. This is the point where PB3 switched back to offering Bicycle A. The lower chart, with prices of Services, also shows two clearly different price levels, corresponding to Bicycle A sharing (lower part) and Bicycle B sharing (upper part). The lines are sometimes interrupted, meaning that the associated PB temporarily stops offering a Service. This corresponds directly to figure 8.1.

The price volatility charts show that prices in the model can fluctuate, but remain within a certain bandwidth: the 'market equilibrium bandwidth'. Within the bandwidth, businesses have some freedom to increase prices without losing loyal customers. However, if the price difference with competing offers on the market becomes too large, customers will switch away from PBs that do not adjust their prices downwards.

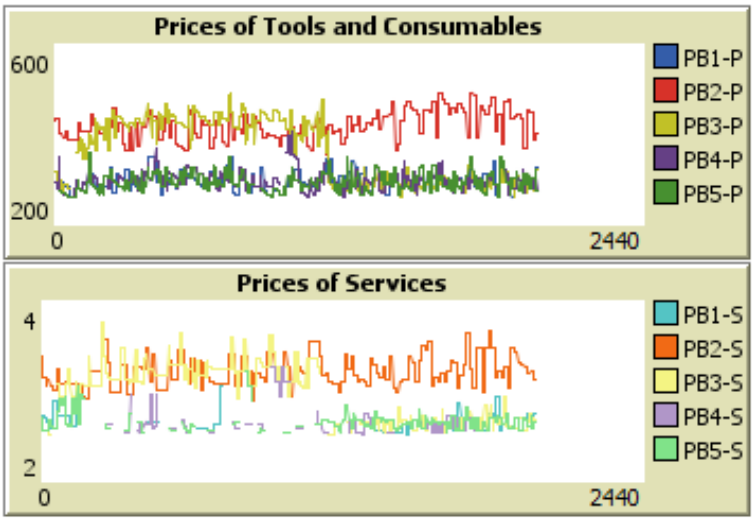

Figure 8.2: Prices of Products (first chart) and Services (second chart) over the course of one simulation run

On the other hand, if PBs collectively increase price levels, at some point one of them will find out that dropping her price will attract so many CAs that the volume makes up for the lower margin, resulting in a higher profit. Other PBs then have to follow with low prices as well and the whole cycle can start again.

Figure 8.3 shows how the market shares of each SM (the product-based and service-based SMs of each PB) fluctuated over the course of the simulation run. It shows that Bicycle A had an especially high market share, in particular after the point where PB3 made the switch to the MM associated with that bicycle. Apparently, the initial state of the market did not represent a long-term equilibrium, at least not in this simulation run.

Figure 8.4 shows how the aggregate profits of the PB develop over time. Knowing the choices that the PBs made over time, this chart indicates that Bicycle B was a more lucrative business opportunity than Bicycle A in this simulation run. PB3 switched from B to A somewhere halfway, but should not have done this as it apparently resulted in far lower additional profit per time unit. The choice was probably made because of some unfortunate market research, i.e. in the random selection of the market sample relatively many CAs were chosen with an effective preference for Bicycle A. Moreover, Bicycle A may have been a better fit with the strategic preferences of PB3.

Apparently, later strategic reconsiderations of that PB never indicated that a switch back to Bicycle B would yield sufficient additional profit to justify switching costs. The chart shows that switching costs can be substantial: the costs of switches of PB4 and PB5, at approximately one third of the simulation runtime, caused a significant setback in their net profits. The profits of PB4 were even temporarily below zero.

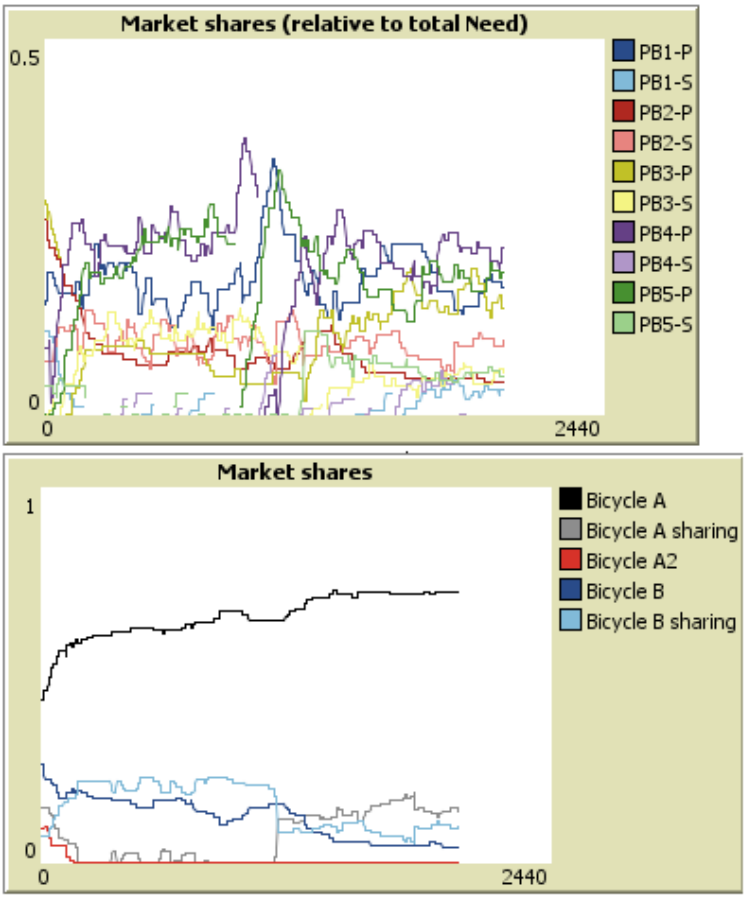

Figure 8.3: Market shares of each SM (upper chart) and each Product and Service (lower chart)

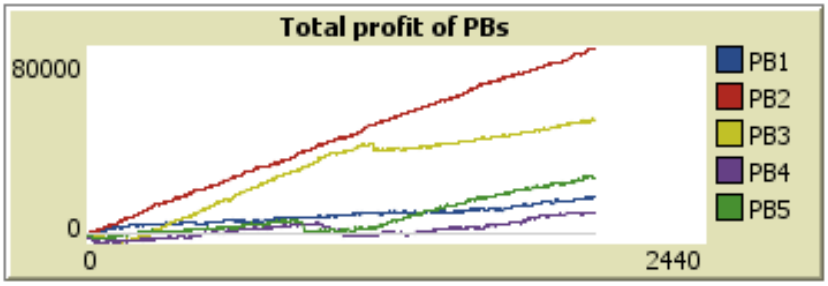

Figure 8.4: Profits of each PB over the considered simulation run

Figure 8.5 shows that, despite the fluctutations in prices and market shares shown above, the overall fraction servicised remains remarkably stable. Figure 8.6 indicates that the same is true for the use of resources per time step, but that associated emissions of CO_2 and CH_4 depend a bit on the relative share of

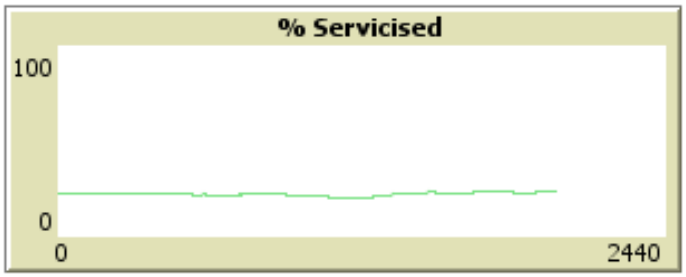

Figure 8.5: Fraction of consumption servicised over the considered simulation run

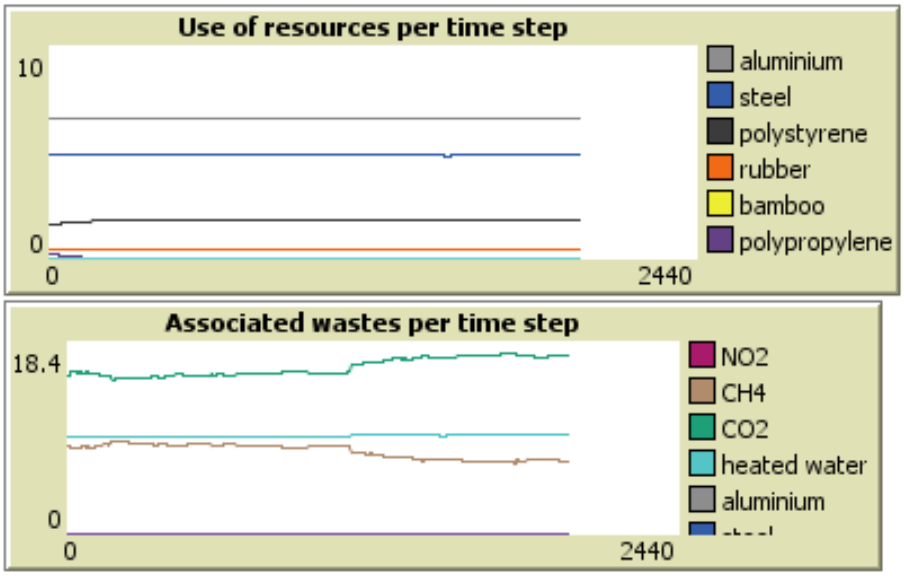

Figure 8.6: Environmental effects over the considered simulation run

consumption that is based on Bicycle A or Bicycle B.

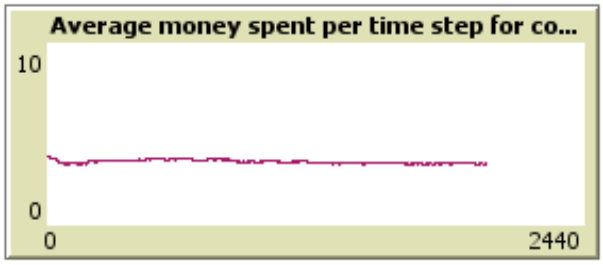

Figure 8.7: Average amount of money spent on consumption over the considered simulation run

Figure 8.7 shows that regardless of the exact composition of offers, CAs pay approximately the same

amount of money for fulfilling their Need over the whole simulation run. However, this is an aggregate statistic, inspection has confirmed that individual values show more volatility.

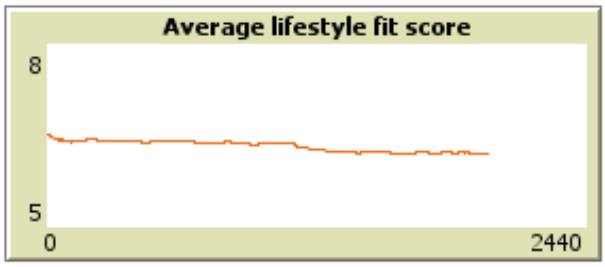

Figure 8.8: Average lifestyle fit over the considered simulation run

Finally, figure 8.8 shows that the lifestyle fit decreases over time, as a larger part of consumption that was previously based on Bicycle B shifted to consumption based on Bicycle A (which generally has lower scores on preference aspects, resulting in a lower 'fit' value for all agents).

8.4 External Influences

This section explores the effects of three possible policies targeted at stimulating servicising in the bicycle market. Some of these policies are somewhat overdimensioned, but this helps in observing the type of effects they can have. All three Policy Instruments are modelled to activate at time step 250. The first two policies (related to subsidies and taxes) have temporary effects, the third policy (environmental awareness campaign) has a permanent effect.

Please keep in mind that none of these findings contain any reliable information on real bicycle markets or on servicising shifts in general. The exploration of fictive policy merely illustrates the types of analysis that the model supports.

8.4.1 Subsidy for bicycle sharing

The first explored policy provides a subsidy of 1 euro for every sold hour of bicycle sharing. The hypothesis is that this will cause a substantial shift towards bicycle sharing. When the policy deactivates, some PBs will probably switch back. Part of the CAs may then also switch back. However, since certain investments in servicising have been made, the post-policy level of servicising is expected to be higher than it was before the policy activated. Figure 8.9 shows the actual simulation results.

The results indicate that the subsidy has caused an absolute shift towards servicising for the duration

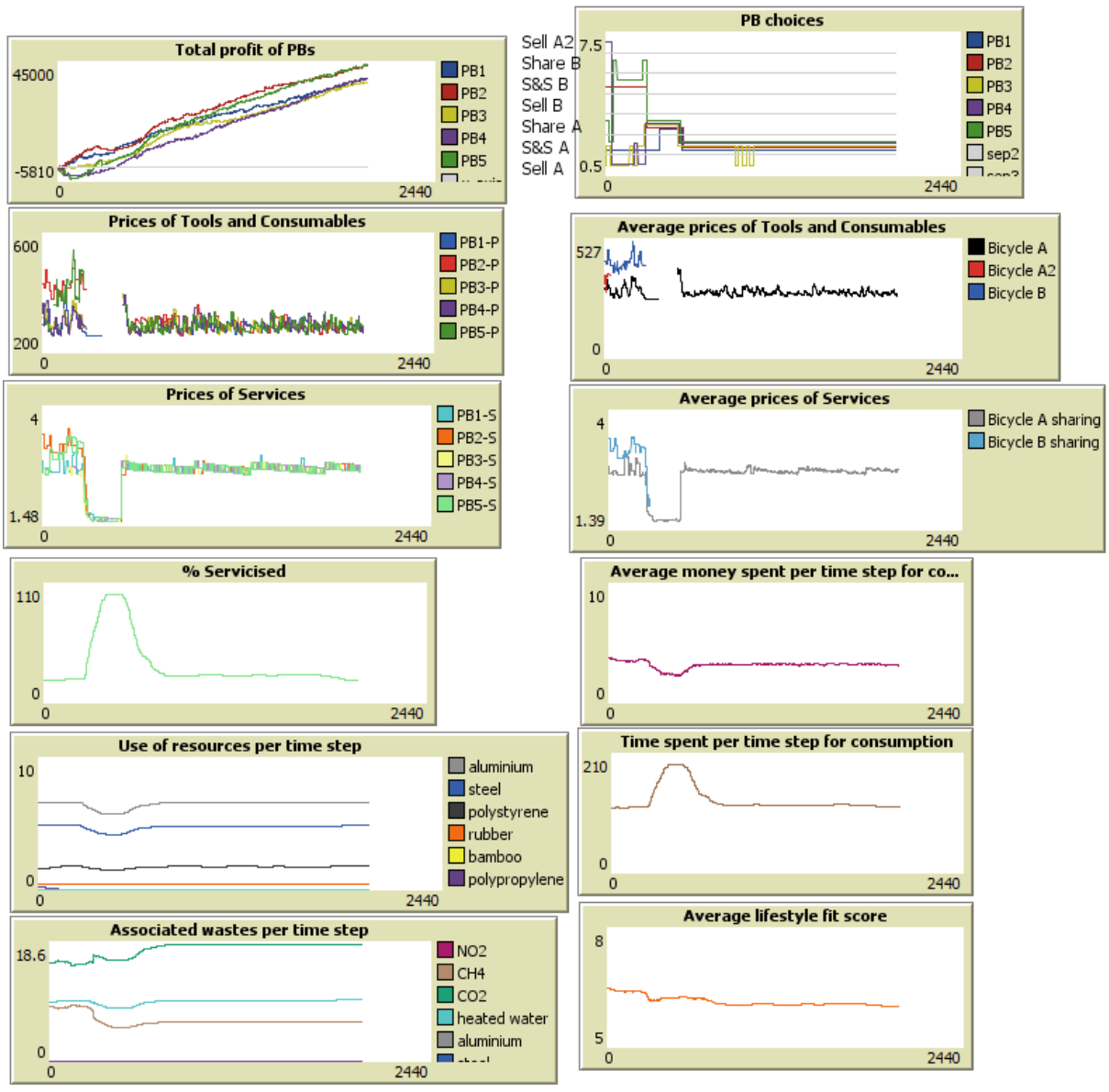

Figure 8.9: Model outcomes with activation of a subsidising policy between time steps 250 and 500.

of the policy. When the policy deactivated, however, the service-based fraction of consumption returns to almost the same level as before the policy. The environmental impact graphs show that the servicising period resulted in lower use of resources, but still only marginally. This is probably a result of parametrisation.

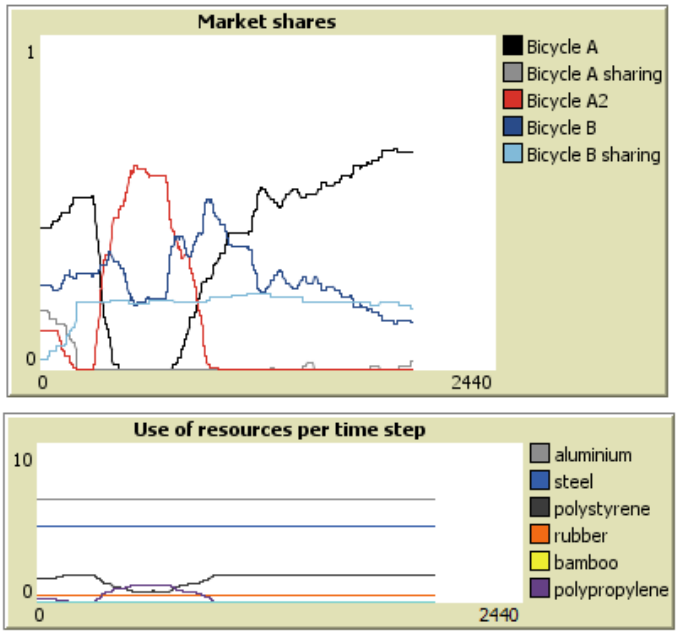

Figure 8.10: Market shares and use of resources resulting from a policy that taxes polystyrene use between time steps 250 and 500.

8.4.2 Tax on polystyrene

Polystyrene is one of the main components of Bicycle A. Bicycle B uses polypropylene instead. The hypothesis is thus that a substantial tax on plastic A will lead to more use of Bicycle A2, with associated lower environmental impacts. Figure 8.10 shows the relevant simulation results.

The figure shows that indeed a complete shift takes place from Bicycle A to Bicycle A2. This also reduces the use of polystyrene and increases the use of polystyrene as expected. However, when the policy deactivates, the switch is completely reversed. Apparently, the switching costs were still low enough (and Bicycle A attractive enough) to switch back.

8.4.3 Environmental awareness campaign

The third and final fictive policy explored in this thesis comprises an environmental awareness campaign. The campaign increases the weight that CAs put on environmental aspects by two points. As we use a five point scale, the weight can never exceed five. Since services have been parametrised with a better environmental score, but many other factors are considered by CAs as well, the hypothesis here is that the campaign will moderately increase the fraction of service-based consumption but will have a lasting effect. This should

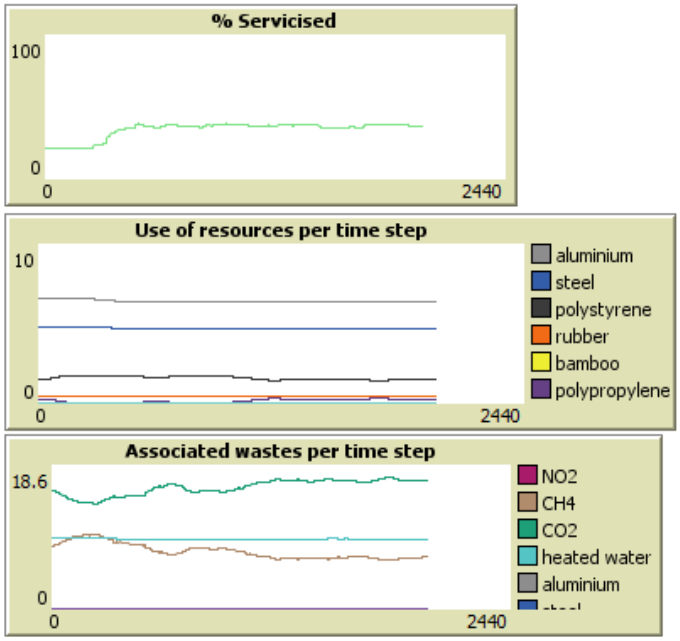

Figure 8.11: Fraction servicised and environmental impacts resulting from an environmental awareness campaign at time step 250

also result in environmental benefits. Figure 8.11 shows the simulation outcomes.

The model outcomes show that the service-based fraction of consumption increases substantially, to a level almost twice as high as before the campaign. However, there is no clearly positive effect on resource use and generated wastes. This has to be a result of the parametrisation of services: apparently they are not as environmentally friendly as one would expect. Since this is a fictive case study, it does not really matter, but in an actual case study this would clearly be something to investigate. Are the services parametrised incorrectly, or do they actually have very little potential to reduce environmental impacts?

8.5 Additional observations

Based on additional extensive experimentation and thorough analysis of model behaviour, I have made some additional observations on model behaviour.

8.5.1 Stabilising forces

Loyalty and market penetration are the main stabilising forces in the model, although their relative influence depends on the parametrisation of CAs. If CAs have high loyalty values, they are clearly less likely to switch

to another supplier unless the price (or preference fit) difference is substantial. Also, if a certain Product or Service has reached a certain market share and many CAs have thresholds or high weights for the market share, then the offer is likely to keep a high market share or increase it even further.

8.5.2 Dynamic forces

Competition between PBs and stochasticity in market research are the two main reasons for volatile market behaviour, especially when looking at PB choices. Competition leads to PBs constantly adjusting prices to optimise their profit given the choices of their competitors. Stochasticity in market research can lead to different perceptions about 'what the market wants', meaning that alternative business models can suddenly appear more profitable.

8.5.3 Problematic situations

Under certain conditions, PBs in the model can set prices of Products and Services unrealistically high. Further analysis has indicated that this is almost always a logical result of monopoly conditions. These conditions are more likely to occur if there are few PBs and depend on the following conditions:

- At least some of the CAs have preference thresholds
- At least some of the offers violate these thresholds
- Only one PB offers the Product or Service with the lowest threshold violation

If all these conditions are fulfilled, the CAs with critical threshold values have only one PB to choose from. During market research, they will indicate that they are willing to pay basically anything (up to their specified budget) for the offer of that one PB. The PB may then set the offer price much higher than would be realistic. This indicates that the current approach to dealing with thresholds may require some modifications. It also cautions the user in its specification of thresholds, as thresholds in the model have a rather strong influence on consumer (and producer) choice.

Small ROI periods (i.e., the time period that agents look ahead for strategic decisions) and small sizes of market research samples can also lead to unexpected and implausible results.

8.6 Provided functionality

Chapter 4 has identified several components of desired model functionality. After the illustration of model behaviour provided in this chapter, it is now possible to evaluate how, and to what extent, the implemented

model addresses each of these aspects.

8.6.1 Basic functionality

The model must represent an artificial market of producers/sellers and consumers. This is clearly the case. There are producing agents and consuming agents. Consuming agents can even be characterised as businesses or individuals, allowing for B2B as well as B2C markets.

The model must simulate the behaviour of market participants over time. Check. Both the producing and the consuming agents have behavioural rules on a strategic, tactical and operational level. As illustrated earlier in this chapter, these rules result in dynamic interaction over time.

The model must allow for competition between products and services that can both satisfy some consumer need. Producing businesses in the model are free to offer products, services or both. The consumer is free to choose for the option that best suits her lifestyle. As a result, the market outcomes show fierce competition. The fictive policies explored in this chapter show that, given the right conditions, it is possible that all product-based consumption gets replaced by service-based consumption. This clearly demonstrates competition between the two.

The model must explicitly address the possibility of a shift towards servicising. The model does this by making an explicit distinction between product-based and service-based sales models, both of which can be based on a shared manufacturing model. Consumption models are also either product-based or service based. Agents are free to choose different models and thereby collectively create a shift towards (or away from) servicising.

The model must be generic enough for application in multiple domains, including at least mobility, agri-food and water use. The generic nature of the model is explicitly discussed in chapter 6. It is designed such that it contains all functionality that may be needed in any of the case studies. Each case study can therefore use the exact same model, ignoring the modules that are not relevant (enough) in the corresponding system. The set of preferences is completely customisable to represent the set of preference aspects most relevant in the market under consideration. Section 8.6.4 will discuss how the model can address some specific requirements of each case study.

The model must facilitate the exploration of economic, social and environmental effects of policy. The model captures economic effects by keeping track of the profits of businesses and the total

spending of consumers. Social effects are mainly represented as the lifestyle fit that indicates how closely a consuming agent's consumption model matches her individual set of preferences. Environmental effects are incorporated by keeping track of associated resource use and generated waste for all material flows in the model.

8.6.2 Functionality relating to business actors

Business agents must have a unique set of beliefs/preferences about how they want to operate in the market. Business agents in the model have such a set and use it in their reconsideration algorithms.

Business agents must periodically re-evaluate both strategic and tactical choices, taking into account their unique beliefs. This is already covered above.

Business agents must consider structural changes in their business models (e.g. adopt a more servicising-based business model if the conditions are right). Business agents are free to choose a different manufacturing model and/or sales model. However, switching costs and strategic considerations may prevent them from doing so. Experimentation with different policies shows that PBs can indeed adopt structurally different business models if the conditions are right.

Business agents must make decisions in a 'smarter' way than just trial-and-error, taking into account things such as market demand and (return on) investment. Producing businesses in the model rely on a sophisticated forecasting capability. This includes market research to get an idea of the market demand for products and services that the business considers to offer. It also includes a full calculation of expected costs and revenues over their predefined 'return on investment' period.

Business agents must not have full information on consumer's preferences, but at the same time not be completely unaware of them. The market demand gives producing businesses an idea of market demand, but is always based on just a fraction of the market. This introduces the possibility that the business agent makes a decision based on biased information. However, in general it will provide reliable information to base her decisions on.

8.6.3 Functionality relating to consumer actors

Consuming agents must have a unique set (although possibly correlated within groups) of consumption needs, personal preferences and susceptibility to social influences that represents

their lifestyle. Consuming agents have such a set. Susceptibility to social influences is included by using market share of a product or service as a proxy for social pressure.

Consuming agents must take the choices made by other consumers into account when making their own choices. As mentioned above, this is included by letting consumers take market shares into account during their decision-making process.

Consuming agents must take into account their own historic choices when making new choices, for example by favouring a previously selected supplier. Consumers have a 'loyalty' property that lets them favour their last selected supplier. Historic choices are not taken into account in any other way.

Consuming agents must be able to filter on products and services based on certain thresholds (including social thresholds), but must also be cognitively capable of making a good choice out of remaining possibilities based on their unique preferences. The model allows consuming agents to have thresholds and weights for each of the preference aspects defined for the modelled market. The thresholds are used to filter options out, the weights are then used to compare remaining options. A willingness-to-pay property allows them to make a trade-off between costs and utility.

8.6.4 Model suitability for the three case studies in SPREE

In the end, the primary use of the SPREE generic model is as a template for three case-specific studies, in the sectors mobility, agri-food and water. This section discusses briefly for each case the servicising shift of interest, the producing and consuming agents and some specific requirements. It then explains for each requirement how it can be represented in the model.

Mobility. The servicising shift of interest in the mobility case is from car ownership to car sharing and similar function-based consumption models for mobility, possibly in combination with public transportation. Producers are manufacturers (or purchasers) and sellers of cars and related services. Consumers are individuals and organisations with a certain need for mobility by car. They can be heterogeneous based on background properties such as family composition and reason for travelling. This will result in different sets of weights and thresholds for properties of cars and services such as the number of seats, storage space, fuel efficiency, design quality and accessibility. People that will use the car for business travel as well as family vacations can have thresholds and weights that take the requirements for both options into account.

The main difficulties in this case are given by the wide range of different cars and the interaction with public transportation. We will define a small subset of different cars, which can be offered as a product

or service (although some cars may only be available in one form). Since the SPREE model only allows consuming agents to have one Consumption Model, we will define composite Consumption Models that represent a combination of car sharing and public transportation. Parametrisation of such models can then be based on an assumption on the relative usage of both.

Agri-food. For the agri-food case, we are mainly interested in the market penetration of integrated pest management (IPM) services, as opposed to farmers buying their own pest protection machines and pesticides. IPM can be much more efficient with regard to used pesticides and invested time and money, resulting in possible benefits for both parties. Producers can be sellers of pesticides and service providers of IPM. Consumers are always Consuming Businesses and represent farmers with a need for protection against pests.

Important concepts in this case concern the relations between skill, costs, efficiency and yield. There can be multiple types of Sales Model and Consumption Models. Each model eventually provides pest protection, but with different skill requirements, costs (for both parties), efficiency of pesticide use, and impact on crop yield. Agents are parametrised with certain initial Skills but can learn new Skills through education. If the offer of IPM service providers is attractive enough, more farmers will switch to IPM services, hopefully resulting in lower use of pesticides and higher profits of all parties involved.

Water. The water case is actually not so much about water as it is about water appliances, such as grey water recycling and rainwater harvesting systems. Servicising can help by reducing up-front costs for consumers and offer such systems as a service. Producers in the model will be providers of water appliances, possibly including the water provider itself. The pool of consumers could just be the set of all water consumers in some geographic area. Some of them will be willing to adopt water use reduction systems just because it is good for the environment, while others can only be convinced by significant savings on their water bill. Consumption Models regard the consumption of water (as a secondary in- and output) but have as a primary input the gradual use of (a) water appliance(s).

An important concept in this case study is the existence of water meters. While some consumers pay a fixed amount of money per month for water use, people with meters may pay for the actually consumed volume. This means that we need duplicate Consumption Models for people with and without meters, both with different cost structures. Meters can be represented as Infrastructures, which are then prerequisites for some of the Consumption Models. Users of Consumption Models without a meter will not experience cost reductions, but may still choose this way of consumption because of its 'green' properties.

8.7 Conclusion

This chapter has explored the model behaviour, based on an illustrative case. It has pointed out that the results of simulation runs can differ between each run, which means that a substantial number of runs must be done before one should draw conclusions. The chapter has demonstrated the variety of output graphs that the model can generate, allowing the user to explore economic, social and environmental effects as well as the underlying causes. Moreover, the exploration of fictive policies has demonstrated the potential of the model to explore the impacts of policies of a varying nature. The conclusions based on the illustrative case cannot be used to draw conclusions on servicising in general. However, they show how the model can be used (given sufficient data collection) to meet the objectives in the SPREE project. The last sections of this chapter have systematically discussed how all desired functionality is somehow addressed by the SPREE generic model, and how the SPREE generic model can be paramatrised for each of the three SPREE case studies.

9 Conclusion

9.1 Research structure

This thesis has documented the results of my master thesis project on using agent-based modelling to explore servicising shifts and inform servicising policy.

9.1.1 Servicising

In a world where economic growth is still coupled to increased use of resources, the European Commission is looking for ways to achieve to absolute decoupling. Servicising is a promising phenomenon in this context that can be described as a shift from ownership-based to function-based production and consumption models. As elaborated in chapter 2, servicising can contribute to absolute decoupling of resource use and economic growth, as well as other societal objectives. Despite its potential, servicising is not yet widely adopted. Possible explanations include non-rationality in consumer behaviour (e.g. status-seeking considerations) and a scarcity of viable business models. The potentially beneficial effects may justify targeted policies, but it is unknown which (combinations of) policies will be most effective with regard to the eventual goal of absolute decoupling.

9.1.2 Project goal and research question

The pan-European SPREE research project aims to inform such servicising policies. Within SPREE, I was responsible for developing the generic agent-based simulation model that provides the basis for case studies on servicising policy in three different sectors of the economy. The main goal in this thesis project was to develop a generic agent-based simulation model that can help explore possible economic, social and environmental effects of servicising policies. The agent-based approach is especially useful because of the heterogeneity and non-rationality among consumers and producers that strongly affect their choices (see chapter 3 for more elaboration).

The central research question in this thesis relates closely to the research goal: *What should a generic Agent-Based model look like that can measure social, economic and environmental impacts of servicising policies?*

The model must be *generic* because it will serve as the basis for case studies in different sectors of the economy. The most important desired functionalities are that it represents an artificial market of buyers and sellers, that it supports simulation of the introduction of possible (combinations) of policies and that it provides insights in the economic, social and environmental effects of these policies.

9.2 Main features of the model

9.2.1 Approach

I have chosen a descriptive (as opposed to abstract/simplistic) modelling approach, keeping the definitions of model elements as close to real systems as possible while still being generically applicable to different markets. Rather than just addressing the overlapping set of required functionality for the three planned case studies, the model already includes much of the functionality that may be required in any of them. As a result, the model addresses a large variety of concepts relating to producers and consumers and to their individual behaviour, decisions and interaction on the market. The model adresses a variety of relevant concepts, including business and consumption models, non-rational preferences, market research, infrastructure, skills, switching and production costs and environmental impacts. Chapter 4 has presented a list of desired model functionality and chapter 8 has discussed how each item on that list is addressed in the model.

9.2.2 Overall structure

The SPREE generic model represents a generic market of sellers and buyers that revolves around some central consumption need. Sellers have a business model that allows to sell one type of product and one type of service. Consumers can either represent businesses or individuals, and have a consumption model that specifies how they can convert a product or service offered on the market into their primary need. Both the sellers and the buyers make strategic and tactical level choices with regard to their business/consumption model. Their activities on the operational level result in conversions and flows of products and services. Environmental impacts (resource use and associated waste) are monitored throughout the production and consumption process. Policies and external market developments can change model parameters (including the properties of agents and objects) and thereby lead to different market outcomes. Chapter 6 and appendix

B provide a more detailed specification of model elements and structure.

9.2.3 Agent decisions and interactions

Decisions of agents in the SPREE generic model are primarily made during periodic reconsiderations. The decisions are based on trade-offs between monetary aspects and personal, social and/or strategic preferences. The non-monetary preferences represent the strategic beliefs of businesses, and the lifestyle of consuming individuals.

Sellers use market research to evaluate options for their business model configuration and to optimise their selling price. During market research, they ask buyers at what price they would switch to some new offer. The buyers themselves choose between consumption models to satisfy their primary need, in which they are restricted by the offers on the market.

Products and services have scores on a predefined set of preference aspects relevant in the artificial market. All agents have an individual set of 'preference weights' and 'preference thresholds' as numerical values on each predefined preference aspect. This set represents the strategic beliefs of a business or the lifestyle of an individual. Agents can also have preferences for the current market penetration of the offered product or service.

Decision making of sellers and buyers is very similar. The producing and consuming businesses both look primarily for the option that has the highest expected profit, while consuming individuals primarily try to minimise costs. Both types of agents first filter the available options (i.e. business model configurations or consumption models) based on their preference thresholds. After that, they calculate the total preference fit of each remaining option, i.e. the summed product of scores (specific to the considered option) and preference weights (heterogeneously defined for the agent). Agents are willing to sacrifice a fraction of their profit, or pay an additional fraction of the minimal costs, for options with a better fit. Loyalty to a specific supplier also counts towards the trade-off. The agent chooses the offer with the best combination of monetary aspects and personal preferences.

Chapter 6 and appendix C discuss the main model procedures, including the decision making processes, in more detail.

9.2.4 Model implementation and behaviour

Chapter 8 has explored the model behaviour based on an illustrative case with fictive data on a Dutch bicycle market. Properties of offered products and services can (and will) change throughout a simulation run, and agents react to each others' decisions. The observed market behaviour is dynamic and can vary between

simulation runs. Primarily as a result of competition on the market, agents make different choices over time, leading to fluctuations in prices and market shares. The model shows the implications of these fluctuations on economic aspects (profit, cost), social aspects (strategic and especially lifestyle fit) and environmental aspects (resource use and associated waste).

9.2.5 Most important limitations

Despite its integration of many concepts, the model still has important limitations. For instance, the model can only represent one 'link' in a supply chain, ignoring all upstream and downstream activity. It does not include market entry and leave, nor does it allow for the modelling of C2C markets. Agents in the model have no meaningful geographic position and are not connected through social network structures. Consumers in the model have full knowledge of the market and are unaffected by subjective perceptions. They make decisions based on a linear combination of weights and scores, which is at best a very abstract interpretation of reality. Agents do not have explicit 'learning' processes to help make better decisions over time. The model heavily simplifies business models, ignoring aspects such as financing considerations, diversified portfolios and detailed specifications of contracts and risk. Finally, the model has elaborate data requirements, posing a significant barrier for use in a case study.

With all its limitations, I still think that the model can provide very useful insights about the dynamic interrelations between producers and consumers and the possible effects of policies on these interrelations. Most importantly, it meets the required functionality to be able to support the three case studies planned in the SPREE project.

9.3 Future work: possible model extensions

The model in its current state is already suitable for case studies in many different sectors and not necessarily just for servicising. Many of the models proposed in literature (see chapter 5) can also be specified as a parametrisation of the SPREE generic model. However, as the discussed model limitations already indicated, there is still much room for improvement. A number of possible extensions to the SPREE model can make it more realistic or allow to explore the influence of some additional aspect that could be relevant in certain case studies. This collection of possible extensions paves the way for continued development of the model in the upcoming years.

9.3.1 Small extensions

Throughout the model development process, we have chosen to ignore for the time being the following possible model aspects, although they are relatively easy to implement and may substantially enrich the model:

- Increase in production efficiency over time, due to learning effects (this is different from learning processes discussed earlier).
- Laziness factor: some studies indicate that certain consumer groups may just repeat their previous choice (see also chapter 5).
- Include a set of properties of producing businesses that consumers may evaluate in addition to the properties of their offer. E.g. customer intimacy, brand strength, operational excellence, environmental image.
- Market entry and leave, and associated conditions (when to enter/leave).
- Introduce more bounded rationality, for example that consumers 'know' of only a fraction of the offers.
- Advertisement efforts for new products. This may somewhat counteract market penetration preferences of consumers.
- Contract breach penalties (also during strategic reconsideration and allocation of production volumes). This makes it more difficult to switch away from servicising.
- Adaptive budget of consuming agents, where they gradually adjust their budget to market conditions and are not willing to suddenly pay much more.
- Consumers considering the 'moving average' (i.e. the average over the last x time steps) of prices in their cost forecasts, instead of just the current price.

9.3.2 Big extensions

In contrast to the relatively small extensions listed above, the following extensions require more structural changes of the model elements, relations and/or procedures:

- Multinomial logit. Multinomial logit assigns probabilities based on calculated utility, which may be more realistic than always choosing the best offer. However, the functioning and performance of large parts of the model (especially market research) currently depend on the pairwise comparisons with one

absolute 'winner'. Replacing this by multinomial logit requires structural changes. Also, as discussed in section 5.5.3, multinomial logit introduces the possibly problematic IIA assumption.

- Chains. The model does not currently allow for chains of selling/buying relations between agents. Agents are either sellers or buyers. Making the model suitable for the analysis of behaviour along longer supply chains would require to merge selling and buying logic in one agent type. It would also have implications for market research: intermediary agents cannot easily say what price they are willing to pay, as this depends on their own customers and competition.

- Social networks. Many of the proposed models of artificial markets identified in chapter 5 feature social networks between agents. Such networks may help in explaining, for instance, why certain innovations only reach certain subpopulation. Including this in the SPREE model is entirely possible, but adds a whole new layer of complexity. Also, the networks may prove difficult to populate with data.

- Spatial explicitness. For similar reasons as mentioned for social networks, the model would be significantly enriched by adding geographic information of the agents in the model. Currently, sellers and buyers have no geographically significant location in the model. Such information may help to simulate location-based interactions and policy effects in a more natural way.

10 | Reflection

10.1 Introduction

In this final chapter I reflect on the work done. In section 10.2 I reflect on my experience with using the methodology proposed by Nikolic et al. that I used throughout the model development. I evaluate how well the methodology worked for me, specifically in the context of a *generic* model, and I will try to provide generically applicable recommendations for future modellers. Next, in section 10.3 I will discuss some aspects of my thesis project that make it different from other thesis projects, such as the context of being a graduate student amidst a large European project, and working as a graduate student in a team. Finally, in section 10.4 I summarise the main societal, methodological and scientific contributions of this study.

10.2 Reflection on methodology

The development process of the SPREE generic model followed the methodology proposed by Nikolic et al. in their book on ABMS (Nikolic et al., 2012). The steps they propose result from many combined years of ABMS experience and capture the best practices they have developed. However, their proposed methodology has not been tested much by other modellers, and definitely not for a model of a generic nature.

In this section, I reflect on my experience with using the methodology for developing the SPREE generic model. I will go through each step that was relevant for this thesis, i.e. steps 1 through 6. For each step, I will:

- Provide, for the sake of completeness, the description of the step that was already given in the introduction.
- Reflect on my general experience of carrying out the associated activities in the context of the SPREE project.

- Specifically discuss how suitable the methodology was given the generic nature of the model to be developed.
- Formulate some generically applicable lessons and recommendations for future modellers.

10.2.1 Step 1: Problem formulation and actor identification

Description The need for a model typically derives from a certain problem. The first step in model development is to understand exactly what the problem is, including the exact lack of insight, how the observed pattern deviates from the desired pattern, and some hypothesis on the main causes for that deviation. Also, it is important to understand whose problem it is, which other actors are involved and what the role of the modeller is.

General experience within SPREE Step 1 of the methodology is a highly interactive part of the modelling process. The modeller typically sits together with the problem owner to formulate a shared understanding of the problem.

In the SPREE project, the problem formulation was covered during one of the first meetings between project members, in Sweden. We, the modellers, had read ourselves somewhat into recommended articles on servicising. However, we did not read into published articles on related ABMS studies. As a result, we relied purely on our modelling expertise and improvisation to design model structures on the spot. We depended almost completely on the present domain experts to determine what are the main causes and important patterns in relation to the problem.

In hindsight, the initial hypotheses on main causes and relevant actors formulated during that particular session have already laid important foundations for key model elements and mechanisms. They provided the main basis for all further model development. Although the outcomes were definitely good enough to work with, progressive insight has since led to substantial model changes or could not even be incorporated in the model anymore. Substantial changes required substantial time investments and were not always as easy to communicate with other participants of the problem formulation session.

I think that the initial problem formulation could have already been much better, reducing the need for large changes later on. We, the modellers, should have educated ourselves in advance on comparable studies on artificial markets. We should have made an overview of the types of influences, mechanisms and system behaviour that are typically mentioned in those studies as the important drivers of emergent patterns.

In the case of SPREE, such an overview would have provided a useful source of inspiration and discussion. It would have allowed us to introduce, very early in the process, some interesting problem aspects that might

also be relevant to servicising but were not mentioned by the problem owner and thus excluded from early discussions. Examples of such aspects that were long ignored include consumer networks, the difference between actual and *perceived* properties of products and services, and learning processes of both producers and consumers. These were never identified as possibly being important and were therefore not included in early (or later) model versions, although they are central concepts in many other models. If they would have been part of the initial problem formulation sessions, we would at least have had better ground for excluding them and may even have attempted to incorporate some of these aspects as fundamental model features.

Specific observations for a generic model Especially when making a generic model, it is important to consider all (or as many as possible) concepts and mechanisms that may be relevant to the central problem formulation from the start on. Otherwise, the modeller's thinking about the model may remain too close to specific examples, making it impossible to design a coherent, truly generic model that can be applied to multiple domains. Therefore, especially in the case of generic models, the initial identification of relevant concepts and mechanisms is of critical importance and should not rely solely on the (mostly example-based) views of the problem owner.

Lessons and recommendations Based on the experience within SPREE, I can formulate a main lesson with regard to step 1 of the methodology. The initial identification of relevant concepts and mechanisms in relation to the problem formulation is of critical importance for early (and thereby for overall) model development. The suggested approach by the methodology currently depends too much on the problem owner's views and understanding of the system.

I therefore argue that the modeller should actively prepare for problem formulation sessions by already finding out which concepts and mechanisms are considered important in other relevant ABMS studies. This additional information will make these sessions much more effective. It enables the modeller to put the suggestions of the problem owner in the perspective of alternative options, which is especially important when the aim is to design a generic model. This additional effort at the start will also reduce the number of iterations required in the modelling steps following the problem formulation.

10.2.2 Step 2: System identification and decomposition

Description With the problem identified, the next step in the methodology is to make an inventory of all concepts, actors and objects, behaviours, interactions, flows and properties that may be relevant to the problem. The modeller should then structure all the identified elements and the relations between them. Initial structuring is followed by an iterative process of simplification and elimination where possible, such

that only the essential elements remain. The outcome of step 2 is a clear, structured set of elements. In the case of the SPREE model, it provides answers to questions such as:

- Which agents (e.g. producers, consumers) and objects (e.g. products, services) are essential?
- What decisions (e.g. change price, switch to a different consumption model) do the agents make and on what basis (e.g. price vs 'quality', minimum requirements)?
- How do they interact (buying/selling, peer influence)?
- What are the material flows in the model?
- How should policies and external developments be taken into account?
- How can we actually measure economic, social and environmental effects?

General experience within SPREE This step of the methodology consists of two phases: inventory and structuring. In SPREE, the inventory phase of the system identification step was again conducted largely in collaboration with other SPREE participants, with them taking a role similar to that of a problem owner. Since the inventory phase is mainly a brainstorm activity, the collaborative approach was very effective and useful to collect much information from the 'problem owner'. It resulted in a mindmap that already identified many relevant elements, categorised in ABMS terminology but still without much internal cohesion.

Then came the second part of the system identification step: the structuring phase. It was clear (and always had been the intention) that not all identified elements from the inventory phase could be included in the model. A rigorous process of elimination and scoping was needed. The book by Nikolic et al. suggests an iterative process of elimination, questioning the right to existence of each identified element. However, it then became painfully clear that we did not really have a basis for making these decisions. The problem formulation itself provided some help, but was too high-level to make well-founded choices.

As a result of this lacking basis for the structuring process, we frequently had to discuss the specific choices (i.e. which elements to include/exclude) with other SPREE participants. Since they did not always have a good overview of the whole model, it was often challenging to communicate all implications of the choices. In the end, the structuring (and elimination) phase of the system identification came down to a lengthy process of less-than-efficient communication cycles. I think that eventually, we were able to make mostly the right choices, but in an unstructured and inefficient way. Usually, prioritising took place by relating to some central concept or important model functionality, although these were never formalised as such until I did so in chapter 4 of this thesis.

Specific observations for a generic model Domain-specific models have the advantage that most observations of system elements and properties can be directly translated into model components. Generic models do not offer this luxury. They require the modeller to first determine the similarities between (elements of) several domain-specific systems that the model must address. These 'similarities' must then be translated to precise model formulations.

For generic models, the system structuring phase therefore consists not just of elimination and simplification, which will already result in a more elaborate system inventory because there is inspiration from multiple domains. It also consists of a simultaneous process of *generalisation*. This part of system identification is not mentioned at all in the book by Nikolic et al., which appears to implicitly assume that the system of interest is a concrete, specific real-world system.

Lessons and recommendations In my experience during SPREE, the generalisation aspect was the main reason why the structuring phase of system identification needed so much time, discussion and iteration. We had to simultaneously reduce a vast set of possible model elements, without any clear basis to do so, and generalise these elements for applicability to multiple domains. In my opinion, the methodology as described in the book by Nikolic et al. does not adequately help the modeller in the system structuring phase, especially when the model to be designed is of a generic nature.

I propose a more structural approach to the process of elimination and simplification that should simultaneously facilitate the task of generalisation. My proposal is to add, between the inventory and structuring phases, two additional phases within this methodology step that should support the structuring phase and help with structured model development in general. Both steps need to be performed in collaboration between the modeller and the problem owner. Figure 10.1 shows how the two new steps are situated between the inventory and structuring steps.

Step 1:	Problem formulation and actor identification *+background research by the modeller*
Step 2a:	Identification of important concepts *Based on the problem formulation, background research and discussions*
Step 2b:	Identification of desired model functionality *Based on the problem formulation and identified concepts*
Step 2c:	System identification and decomposition *Inventory + structuring, based on concepts and functionality*

Figure 10.1: Proposed additional phases of step 2 of the methodology

First of these two steps, the modeller and problem owner should create a shortlist of central/critical model concepts, preferably ranked by order of importance. The list should be strongly related to the

problem formulation, which in turn should be fueled by the modeller's research on comparable studies as recommended for step 1. Dominant findings during the system inventory phase can also be taken into account as a baseline. In chapter 4 of this thesis, I discuss how I derived such a shortlist of concepts from the problem identification and findings from literature. This process in itself covers part of the required generalisation activities and provides a basis for further generalisation choices.

Second, the modeller and problem owner should also identify desired model functionality, again preferably ranked by order of importance. The resulting list should again relate to the problem formulation and literature research, but also to the list of important concepts. Chapter 4 of this thesis includes the list of identified model functionality for the SPREE model.

The lists of important concepts and desired model functionality can make the choices during system structuring much easier. They function as a reasoning framework, that allows the modeller to consider for each element under consideration whether it has a direct relation with the concepts and/or functionality. If not, it is probably not necessary to include the element in the model. If the choice is between two elements, the element that relates most directly to the core concepts and functionality should probably be picked (or the lists of concepts and functionality need to be revised).

10.2.3 Step 3: Concept formalisation / the ontology

Description Although the modeller already makes important modelling choices in step 2, this does not yet extend to all the *relations* between elements. Also, the elements are still defined in a natural language that may be context-dependent and ambiguous. Such ambiguity is undesired in general and specifically because computers cannot deal with it. Step 3 therefore serves to relate the elements together and make the elements and relations from the previous step 'computer-understandable' in addition to human-understandable. The methodology proposes two options but encourages the modeller to create a formal 'ontology' that is basically a structured dataset of all model components and the relations between them. The ontology can be a very suitable basis for object-oriented software implementation.

General experience within SPREE When I look back at my own experience during the SPREE project, the process of creating an ontology can be characterised as an iterative combination of structure and chaos. We (the modelling team) started with a big 'box' of elements and concepts that we wanted to include based on the system identification and decomposition step. However, there was still little structure between them. We had many ideas on possible relations and model procedures, but they were ill-defined and incomplete, as we discovered during the formalisation process.

We documented all progress on the model in a central, publicly accessible 'wiki', such that other SPREE

participants could easily follow the progress. For each model element we reserved a separate page, listing its properties and behaviour. Although very useful indeed, such a wiki does not give anyone much insight in how the model elements relate together - it is merely a collection of information on individual model elements. Since it does not provide a good 'helicopter view' I would not really call this an actual ontology.

Looking closer at the conceptualisation process, each day we worked on one (or several) of the elements, relations and procedures. We said a lot of useful things, noted some of them down in the wiki and made some quick diagrams to structure our thoughts. However, we always ended with new questions, problems and exceptions and seldom felt like it was good enough to elaborately write down. Even when we thought something was good enough, a week later we would encounter some related element or procedure that required us to reconsider everything we decided on before. By that time, we had often forgotten half of our original discussions and arguments, which were typically of a very detailed and complex nature. As a result, we went through some of the same thought cycles twice or more.

All in all, the conceptualisation phase was not as efficient as we would have hoped. For some part, this is because we did not make enough notes, nor did we structurally maintain supporting diagrams. However, our experience was also that the book by Nikolic et al. provides a rather vague definition of an ontology, and little to no help for developing an ontology based on the list of system elements identified during step 2. It is clear what the ontology should 'do' and why, but not *how* it can be developed from the identification of system elements that results from step 2. This leaves the modeller with loads of data and no idea where to start from there.

The book refers to another book that 'provides a solid introduction' to the field of ontology methodologies, but does not summarise some of the main conclusions or guidelines. It mentions a number of tools that can be used but does not advise which tool is useful in what situation. Moreover, the book does not suggest any concrete steps that the modeller can take. It does not provide concrete examples of how to unite in one transparent and coherent ontology the properties of all elements of a complex model, as well as the relations of interaction between them. The one example of an ontology in the relevant chapter of the book is basically a simplified UML-diagram. That diagram is too small to understand how a complete ontology can be made for a complex model without turning into an incomprehensible spaghetti model.

Specific observations for a generic model The observations on the ontology discussed above apply equally to domain-specific models and to generic models. However, our ambition to include all concepts in the model that may be relevant in just one of the case study means that the number of elements and relations is quite large. This makes it increasingly difficult to create a transparent ontology and increases the need for adequate tools.

Lessons and recommendations I think that a methodology book that promises to be hands-on should provide the modeller with more concrete guidelines for developing an ontology. It is currently very unclear how a transparent and coherent ontology can be derived from the outcomes of system identification, particularly when the model is large and complex.

Without going into literature on ontologies, which I consider outside the scope of my thesis work, I think that the wiki-based approach used for SPREE can serve as a very useful basis. The book should maybe advocate this specific approach and provide the modeller with a stepwise, hands-on approach to developing an ontology:

1. Make a separate page for each model element, with placeholders for three types of information:
   - Properties/states of the element.
   - Actions it can do or undergo in the model.
   - Relations it has with other model elements.
2. Fill in all known information and come up with the rest.
3. Complementary to the information in the wiki, create visualisations (e.g. in MS Visio or similar software) of the relations between model elements, and maintain them as 'living' documents that are kept up-to-date as new changes are made to the model. The diagrams should correspond directly with the specified relations in the wiki pages. The three diagrams that I provide in chapter 6 may turn out to be quite generally applicable: one for agent-agent relations, one for object flows and one comprehensive agent-object-object relation diagram. Results of discussions should be directly incorporated into the diagrams whenever possible.
4. Iterate, also in collaboration with the problem owner, until satisfied with the result.

In addition, the book should emphasise on the following good practices that should make the ontology development process more efficient:

- Twice per day, take the time to note down results of discussions or thought processes during the last couple of hours. The wiki can be a very suitable tool for this. Do not just create an ongoing list of comments, but structure them according to the model element that they relate to (possibly even with a substructure per element).
- Create a log or timeline of important decisions. Always summarise detailed and complex discussions and structure the arguments used, possibly in hierarchical visual representations.

The recommendations admittedly introduce a lot of overhead, but in the end they will make the concept formalisation process not only more structured but also more efficient. The current status of the development process can directly be communicated with others and the modeller(s) can revisit previous discussions much more smoothly.

10.2.4 Step 4: Model formalisation / narrative and pseudo-code

Description With the 'who' and 'what' specified in a formal ontology, step 4 requires the modeller to specify who does what and when. The first subtask here is writing a model narrative that describes (in a natural language) the process that agents 'experience' in the model. For example, one phrase in the narrative could be "The agent wakes up, gets a cup of coffee and checks if she still has an option to satisfy her need for mobility this day." The second and last subtask of step 4 is to translate the narrative into a pseudo-code, which is essentially an algorithm written in human-readable form. It forces the modeller to think about a logical model layout without being distracted by software implementation details. Pseudo-code consists of computations, value assignments, iterations, loops, conditions and input/output operations, among possible other things.

General experience within SPREE For the SPREE generic model, we have worked with several representations of a model formalisation. I have personally spent at least two weeks working out all activities identified in the ontology as pseudo-code on a more concrete level. The pseudo-code distinguishes model procedures and their in- and outputs. Also, it relates envisioned model behaviour to model elements and their properties/states. The process of writing pseudo-code constituted an essential step towards model implementation and led to many refinements. Regrettably, the methodology does not offer much advise on how to make the decomposition of multi-layered model procedures more transparent and comprehensible for large models. The resulting pseudo-code was thus a long list of procedure descriptions, with little internal structure.

In parallel and corresponding directly to the pseudo-code, we developed a flowchart that illustrates the hierarchy of procedures and subprocedures. Appendix D contains the latest version of this flowchart, which was kept up-to-date during model development. The flowchart was relatively straightforward to create and maintain, and helped much in keeping an overview of how all the procedures interrelate.

In addition to the pseudocode and the flowchart, we have attempted to write a narrative of agent interaction. However, some procedures in the SPREE model are based on as many as seven levels of subprocedures, all of which are relevant to understand what is happening. This deep complexity (see also section 7.2.1) makes it very difficult to write a narrative that discusses agent activity in sufficient detail while still being

comprehensible. The examples that accompany the methodology description in the book by Nikolic et al. regard much simpler models than the SPREE generic model, with few layers between the top level and the lowest level procedures. This is understandable from a didactic perspective, but it ignores the possibility of more complex models.

Chapter 6 contains a model description that is also close to a narrative, but with an 'observer' perspective instead of an agent perspective. The chapter describes in a structured way the different things that can happen in the model. This is a bit different from a narrative but a more suitable format to explain the core model processes.

Finally, appendix C contains yet another representation of model procedures. It discusses the procedures in a text-based format with pseudo-code elements that can be characterised as somewhere in-between a narrative and a pseudo-code. This format seemed best to communicate in detail the model procedures to interested readers with no programming affinity.

Specific observations for a generic model Relating more specifically to the generic nature of the model, we found that narratives are very difficult to write for generic models. This is especially true if part of the model functionality is optional. The strength of a narrative is that it makes very concrete what agents do. This becomes very difficult if it is not yet known what type of entities the agents represent and whether they will actually do everything that they could possibly do according to the broadly defined results of the conceptualisation phase. I therefore think that a narrative in the form that Nikolic et al. present is mostly suitable for KISS-based (see section 5.7, domain-specific models only.

Lessons and recommendations There are multiple ways of representing the model formalisation and I have experimented with several of them. Based on my experience, I would say that a flowchart of model procedures is the most useful and important technique, and should be used for any type of model. Surprisingly enough, the flowchart is not mentioned at all in the book by Nikolic et al., so my first recommendation with regard to methodology improvement is to add a section on flowcharts.

Complementary to the flowchart, the modeller needs to describe for each model procedure which agents are involved, what they do and based on what information. Pseudo-code can be useful to already think about complex procedures without bothering with implementation details. However, I do not think the modeller needs to write pseudo-code for all procedures in the model. Many 'blocks' in the flowchart are probably rather self-explanatory and can be written in actual software code at once to save time.

A narrative can be useful to explain agent activities in the model to other people. However, deep complexity, generic definitions and optional model functionality all make narratives more difficult to write

and harder to understand. A modular description of important model procedures, similar to what I did in appendix C, may then be a better alternative.

As a final recommendation here, the experience of the SPREE project has taught me the importance of keeping model procedures as *modular* and *generic* as possible. This will keep complex models manageable and understandable, during the model formalisation but also during the model implementation and later phases. In my opinion, the methodology should include some recommendations on these aspects of good modelling practice:

- Modular procedures. Each procedure must have a single purpose. Blocks of codes that are used in multiple procedures should be defined as a separate procedure. This will make individual procedures more understandable, better communicable and easier to debug. It also facilitates reuse of procedures in other procedures (or even in different projects) and allows for modular change of procedures, e.g. replacement of the central decision algorithm by a different approach.
- Generic procedures. All procedures should be defined in a way that is as context-independent as possible. Context-specific variables should be defined as input arguments. For instance, the SPREE model has a separate 'calculate-preference-fit' procedure, that takes as input a list of weights and a list of scores rather than referring directly to the 'strategic-preference-weights' or 'lifestyle-preference-weights' attributes of agents. This makes the procedure cleaner, easier to debug and easier to reuse for another purpose.

A disadvantage of using modular and generic procedures is that they generally take more time to design. However, it is mostly a matter of practice: once you are used to defining procedures in this way, it becomes standard practice and at some point will take virtually no additional time compared to a more straightforward (short-sighted) approach.

10.2.5 Step 5: Software implementation

Description Several good modelling environments are available for building agent-based models, including NetLogo (open source), RePast (open source) and various commercial platforms. Good practices during the software implementation step include version control, elaborate code documentation, naming conventions, division of tasks and responsibilities, and bug tracking.

General experience within SPREE The methodology includes several recommendations for good practices during the model implementation phase, all of which we have adopted and all of which contributed to

a better and more efficient process. Version control helped with synchronisation between model developers and different machines, and allowed us to retrieve accidentally deleted code fragments. Extensive documentation (in the form of comments within the code) was especially important because of the many nested procedures. Without documentation, even the modeller himself forgets about the exact functionality of each model procedure. Also, the writing of code comments forced us to have a second look at each piece of code after it had been written, providing a good initial verification check. Naming conventions (see appendix E) helped us in distinguishing local variables, agent properties/states and global variables. Also, we used naming conventions to remind ourselves when variables store data in the form of lists or tables.

Although model implementation was not divided over multiple teams, within our team of two we did have two different roles: I wrote most of the code and my colleague constantly checked for any mistakes, duplicate code and, for instance, violations of naming conventions. We both experienced this as a very efficient division of tasks and I strongly recommend other programmers of complex agent-based models to always work in pairs. If both team members have substantial programming experience, they can switch roles every now and then.

The software implementation took 2.5 months, which was somewhat longer than anticipated. The main reason for the delay was that I underestimated the work related to model initialisation. As described in chapter 7, we developed an illustrative case study that we used to verify model behaviour after each phase of our implementation roadmap. However, before we could start with the first phase of the roadmap, we had to 'write' (i.e., program in software) the functionality to read in all input data and create model instances from it. Also, these instances had to be visualised in a clever way that facilitated fast inspection and the signalling of potential problems. This process took much more time than anticipated.

In my experience, the remainder of the model implementation process was quite efficient, due to a number of adopted good practices. The roadmap and illustrative case, that together allowed for intermediate verification checks, helped tremendously in keeping our progress structured and catching implementation bugs as early as possible. Working as a pair helped to stay sharp. The NetLogo language allows to program complex interaction patterns in relatively little time. Of course, it also helped that I had previous experience with NetLogo. If neither of us had used NetLogo before, I am pretty confident that we would have either stranded in early model implementation or had needed at least twice as much time until the entire model was finished. In the end we were able to implement a very large model (compared to most NetLogo models) still relatively quickly.

Specific observations for a generic (and large) model The explicit objective of building a generic model also has implications for the implementation phase. It already helps a great deal if the model pro-

cedures have already been formalised in a modular an generic way during step 4. During implementation, the modeller especially has to take care to keep all procedures context-independent, allowing them to run regardless of the model population.

Although relating less to the generic nature of the model than to its size, it is also worth making some remarks on the implementation of such an extensive model in NetLogo. As mentioned already in chapter 7, NetLogo is not really meant for large models but rather for education purposes and quick prototyping. Nonetheless, we went ahead and built the entire model in NetLogo. In doing so we encountered a number of challenges relating to structured code, user friendliness and flexibility:

- NetLogo code is no structured, object-oriented set of modular code files. Instead, as can be seen in appendix E the model code is basically one long text file. This leaves it completely up to the user to introduce structure between the several procedures.
- Navigation within the model code is not supported by any automatic in-text hyperlinking functionality that would allow to directly jump between procedures and/or agents.
- User-defined local variables, agent states/properties, global variables and model procedures are all shown in the exact same font.

As models grow larger, this makes it increasingly difficult to keep an overview of model procedures and how they interrelate, until at some point it becomes simply impossible for one person to comprehend. The primary resulting problem is that bugs can no longer be resolved in that case. I think that our SPREE generic model is already close to the boundaries of model size still manageable without a more sophisticated editor, particularly because of the multi-level complexity of nested procedures. Consistent naming conventions, indentation practices and sufficient documentation allowed us to get this far, but I do not think I could build a model twice as large still in NetLogo. This is not a result of the NetLogo language itself but simply because of the limitations of the editor.

A final limitation of NetLogo that we encountered is that it does not allow the modeller to build custom interfaces. The modeller is completely dependent on standard visual components and buttons, although extensions can add more flexibility. This facilitates quick prototype models but becomes a nightmare if one desires to create a custom user interface without green and purple buttons. Such tasks are definitely much more doable in other software environments.

Lessons and recommendations I cannot stress enough the importance of structured and consistent practices during model implementation. The book by Nikolic et al. contains some very useful recommendations with regard to version control, extensive documentation and naming conventions. I propose the

following additional contributions to the list of good programming practices that should help modellers with the implementation of their formalised model:

- Model implementation should follow a roadmap that consists of multiple phases. The first phase is the most basic version of low-level model interactions. Each following phase adds model functionality, until the final phase includes all planned functionality. Before continuing to a next phase, the modeller should verify that all specifications of completed phases are implemented correctly. This approach helps to keep the implementation process very structured and it facilitates bug tracking.
- The modeller should develop an illustrative case study that includes a full model parametrisation. Such a case study helps significantly to facilitate testing of procedures, understand why bugs occur and communicate about the progress. Also, it allows verification of model functionality at each implementation phase. If the case study is somewhat realistic, this helps in understanding model behaviour more intuitively.
- The modeller should not remove the - usually temporary - lines of code that generate output lines to support debugging. Instead, these lines should be made optionally available. This facilitates future bug tracking and supports verification and validation processes.

With regard to the implementation platform, I can heartily recommend NetLogo as an efficient modelling environment, especially if the model to be developed is not too large or complex. The larger the model to be developed, the more important it becomes to systematically apply all recommended practices. Alternative platforms may then be more suitable as they typically provide more structure as well as more mature environments for code editing.

10.2.6 Step 6: Model verification

Description Before moving on to model use, the modeller should verify that he or she 'built it right', i.e. correctly translated the conceptual model into model code. The methodology proposes several methods for model verification, such as the recording and tracking of agent behaviour, single-agent testing, interaction testing and multi-agent testing. The verification process typically consists of testing hypotheses on agent and model behaviour, given certain inputs or circumstances. The more tests the modeller carries out, the more certain he/she can be that the model was built right.

General experience within SPREE One of the things I think we did very well is the continuous process of verification between two phases of the implementation roadmap. In practice, we largely integrated the

implementation and verification processes. The verification tests suggested in the book by Nikolic et al. are all very useful and the modeller should take some time to carry them out once the model is completed. They can serve as a 'proof' of correct functionality and thereby increase confidence in the model, not just of the modeller but especially of the user. However, in my opinion the continuous comparison of expected and actual behaviour of the model in each intermediate state of implementation is even more important and perhaps the only way to find certain bugs in highly nested procedures in time.

Specific observations for a generic model One cannot properly test a generic model without a specific parametrisation. In our case, the 'Dutch cycling' illustrative case study constituted our test environment. We attempted to make the case study elaborate enough to capture all possible model elements and possible variations. Still, the illustrative case study is only one possible parametrisation of a model that it intended to be generally applicable. Based on the test results with the illustrative case study I cannot guarantee that no new bugs will pop up with a different parametrisation. However, the verification tests that we have done have definitely increased our confidence in the correct functioning of important procedures. I am confident that other case studies will not lead to any major new problems.

Lessons and recommendations My main recommendation for other modellers with regard to model verification is to adopt a process of continuous verification, based on phased model implementation. Our experience with the SPREE generic model has proved the merits of this approach. Additional verification checks are especially useful to build confidence once the model is done. Of course, the techniques can also be used during the continuous verification process.

10.2.7 General findings on methodology

Based on the discussions above, I can make some general observations on my experience with step 1-6 of the methodology proposed in the book by Nikolic et al..

Valuable structure. First of all, the methodology consists of logical steps, relating closely to classical waterfall model approaches for software design. The questions asked during problem formulation are very important and might otherwise easily be overlooked by inexperienced modellers, eager to 'build something'. The rigorous distinction between conceptualisation, formalisation and implementation helps tremendously in the gradual process towards model implementation. Inexperienced modellers may easily skip steps, jumping too early to the model implementation process. They are likely to find out that certain aspects of model functionality do not work together, require different types of variables or have to be redesigned for other

reasons. The methodology steps basically prevent modellers from making a mess and force them to follow a structured approach. Once the model is implemented, the methodology also provides a useful set of possible verification tests to build confidence with the modeller as well as the user.

Room for improvement. Although the steps provide important structure throughout the modelling process, in my opinion each one of them can be made more valuable by adding additional recommendations to aspiring modellers. The list below summarises the main recommendations formulated above:

- *Step 1: Problem formulation.* The methodology should encourage the modeller more to explore publications on related ABMS studies earlier in the process and introduce the findings as input during collaborative problem formulation sessions. This will make all following model development steps better founded in literature and less dependent on initial 'hunches' of the problem owner.

- *Step 2: System identification.* For large models, the structuring of inventorised model elements can be better guided. My proposal is to let the modeller take a step back after the inventory of elements, and first formulate two lists: a list of most important model concepts and a list of desired model functionality. Both lists relate closely to the problem formulation but are also partly based on the initial system identification results. The lists should help in structuring the inventory of system elements: only the elements that relate most closely to core concepts and functionality should be kept while the others can be discarded.

- *Step 3: Concept formalisation.* The methodology encourages the modeller to create an ontology but provides little help in how to do so. I think that the wiki-based approach used during SPREE provides a very workable basis for such an ontology. In order to keep an overview of model relations, the modeller should develop a set of complimentary interrelation diagrams and maintain them as living documents during model development.

- *Step 4: Model formalisation.* In my opinion, the flowchart is the most useful tool during model formalisation and should certainly be covered as part of the methodology. A more text-based pseudo-code, narrative and/or composite format is needed to work out the procedures in more detail. The methodology should emphasise that model procedures be defined as modular and generic as possible at this point.

- *Step 5: Software implementation.* I cannot stress enough the importance of structured and consistent practices during model implementation. In addition to practices already recommended in the book of Nikolic et al., I propose a specific approach to the implementation process. The modeller should

design a phased implementation process, starting at low-level model interactions. Each following phase adds model functionality, until the final phase includes all planned functionality. Before continuing to a next phase, the modeller should verify that all intended features of completed phases are (still) working correctly. An illustrative case study helps much as the basis for model testing of intermediate versions as well as the end result.

- *Step 6: Verification.* My recommendations for step 5 support continuous verification *during* model implementation. I think that a new version of the methodology book should adopt these recommendations and not present verification as something done only afterwards. The suggested verification techniques are still very helpful, also to build confidence in the final model.

Keeping up-to-date notes of discussions, records of decisions and accompanying diagrams is absolutely essential when developing a complex generic model. The methodology does not (by far) stress enough how important this 'shadow process' is. With regard to visualisation, the three element relation diagrams in chapter 6 may provide a generally applicable format to keep track of relations between agents and objects, as well as material flows.

KISS-oriented. Although I have attempted to point out generally applicable aspects, my evaluation of strengths and weaknesses of the methodology of Nikolic et al. was derived in the context of the SPREE *generic* model. To me it appears that the methodology was written for KISS-based (i.e. as simple as possible in every way), domain-specific models. Modellers of more extensive, complex and generic models will encounter difficulties when using this methodology.

In particular the system identification step (step 2) is poorly applicable to elaborate and generic models in its current formulation. My recommendations for that step will hopefully help in the structuring process of system elements. Looking at concept formalisation and model formalisation (steps 3 and 4), the techniques used in examples in the book are also suitable for small models only and become messy for larger and more complex models. The implementation of such models require additional techniques, such as the proposed roadmap and illustrative case. Without such supporting practices, only relatively small models can be implemented based on the guidelines in the book.

Summarising, the methodology proposed in the book of Nikolic et al. is very suitable for the efficient development of relatively small, domain-specific models. It does not scale well to larger and more complex models. My recommendations will help (inexperienced) modellers to handle larger models and experience a model development process that is less frustrating and more efficient.

10.3 Reflection on process

10.3.1 Being a part of SPREE

Rather unusual for a master thesis project, this project was positioned as a central part of a pan-European project - the SPREE project. In practice, this meant that thesis deliverables had actual (external) deadlines and that the choices I made in my research had to be constantly checked with other SPREE experts. For me, the context of a large European project has mostly added to the experience. Good things included that I could rely on a whole network of international experts on servicising. Also, the knowledge that your model will constitute a major basis for the exploration and recommendation of possible EU-wide policies provides a great motivation.

On the other hand, a graduation project poses different demands than an international, collaborative research project. I envisioned a truly generic and comprehensive model for the SPREE project and often had to make a trade-off between model quality and delivery progress. Quality usually prevailed, but towards the end the pressure of project deadlines forced us to leave some aspects out of the model that we would rather have included. Still, I believe that the resulting model is pretty much the best it could have been, given available time and resources and without delaying the overall SPREE progress. The model provides all the required functionality, integrates a large number of concepts relevant to servicising and is suitable for carrying out the three case studies of the SPREE project.

10.3.2 Being a graduation candidate as part of a team

Another aspect of the process that I would like to discuss here is that most of the work for SPREE was done together with another researcher at the faculty. This was of critical importance for the quality of the model. I strongly believe that developing a complex simulation model should never be done alone, if just because you cannot discuss your ideas about the model aloud and realise how stupid (or brilliant) they are. Working in pairs adds a constant quality check to the work done and I could never have built such a complex model all by myself.

The tension here is, of course, that a master thesis project should be carried out by the candidate himself. When working in pairs, it is no longer clear who did what and the result is always a team effort instead of an individual accomplishment. However, I was still able to make substantial personal contributions to the project.

10.3.3 My personal contribution with SPREE and within the TU Delft team

To illustrate my personal contribution within the context of the larger SPREE project and within the TU Delft modelling team, I will point out some specific model aspects that were mostly my design, and some project tasks where I clearly took the lead.

Business and Consumption Models. The first model aspect that I was largely responsible for is the Business Model (together with the Consumption Model). During early model development, I conceptualised production and consumption processes as combinations of compatible transformation processes with a primary input and output as well as secondary in- and outputs. I distinguished four types of transformations, based on the types of main in- and outputs.

In later iterations, the team of business side experts from Israel pointed out that the model should really allow for partial servicising. To accommodate this, I categorised the transformation processes into the more generic (as they can handle all types of in- and output flows) Manufacturing Models, Sales Models and Consumption Models. The conceptualisation of a business model as a combinations of a Manufacturing Model and two Sales Models allows to have multiple types of business outputs that depend on a shared upstream production process but can be evaluated and manipulated individually. I also conceptualised how these Models may depend on Skills and Infrastructures. Of course, this all happened in collaboration with my research partner, but these were model designs that I largely came up with, and worked out, by myself.

Decision logic. Up to a certain point during model conceptualisation, we did not really consider how agents would make choices in the model, just the type of choices that they would make. During model formalisation, we were forced to formalise how decisions would be made. All we knew was that agents would have to engage in a multi-criteria analysis procedure, evaluating at once multiple characteristics of offered products and services.

At that point, I clearly took the lead in working out some possible decision making structures, which we then shared with other SPREE partners. After some discussion, we collaboratively chose the pairwise comparison approach with weights and thresholds also described in this thesis. In working out the details, I constantly encouraged comparisons with how people (especially ourselves) make choices in real life. I think this has resulted in a decision making structure in the model that is potentially (given the right parametrisation) much closer to reality than the usually much more simplified decision algorithms used in other proposed ABMS models on consumer choice.

Economic reasoning. Another model aspect where I did most of the legwork was the economic forecasting and market research. The whole idea of market research originated from me and was then picked up by our supervisor as a novel and realistic model mechanic that very well fits the unique advantages of ABMS (i.e. heterogeneity, bounded rationality). My partly econometric background helped in working out how businesses can extract critical switching prices from consumers, convert them into a demand curve and then calculate the best price to optimise their profits. Market research is an important aspect of business in many sectors, as is price/demand optimisation. I therefore think that this aspect of the SPREE model provides another significant contribution to the field of artificial market modelling.

Implementation. The last aspect I will mention here as an example of my personal contribution is the model implementation process. Given that I was the only one among the two of use with previous experience in building agent-based models (or programming outside of Matlab at all), I basically did all the model coding. My research colleague provided an essential continuous quality check by looking 'over my shoulder'. In the meanwhile, I was able to share with my research colleague my experience with some of the ins and outs of NetLogo and programming in general.

General aspects. Throughout the entire model development process, I was the only one who constantly had an overview of the model elements we identified so far, how they interrelated, and what the implications were for new design choices. Also, I created many supporting diagrams to visualise model structures, and worked on presentations for our supervisor and for other SPREE participants to enable them to communicate about the model with others. These tasks have no direct relation to the results presented in this thesis, but were still important personal contributions to the project.

10.3.4 Personal choices and time investment

As already mentioned above, project progress was often sacrificed for model quality. These choices were usually the result of my stubborn resistance to accept too strong unrealistic abstractions in the model. The other members of the TU Delft team usually voted for the faster, more simplistic approach. I am still convinced that the latter, more abstract approach would have resulted in too loose a connection to reality. However, largely as the result of my choices, model development took longer than anticipated.

In addition, parallel to the SPREE project I was also involved in some other projects that demanded more of my time than anticipated. Most notably, I worked on a scientific article on another ABMS model that I developed and used over the last couple of years in collaboration with a Boston-based research company. Given the time consumption of this and other side projects, in the 11 months that I have worked on the

SPREE project I spent a net 6.5 months in terms of working days on SPREE and on writing this thesis. This only slightly exceeds the officially prescribed time investment of 5 months.

As a result of both sources of delay (quality over progress + side projects), the first fully working version of the generic SPREE model was not delivered until the end of October 2013 (initially planned for the end of July). Fortunately, we were able to communicate the results from the model formalisation phase efficiently enough to prevent any delay in the overall SPREE project. The inclusion of multiple additional aspects already in the generic model also means that the development of the sector-specific models will take much less time than anticipated and we are back on track with regard to the timely delivery of the sector-specific models.

Despite some of the encountered challenges described above, the end result is a fully functional simulation model. We managed to integrate a complex set of interrelating elements into one generic simulation model. It is true that we did not adhere to the initial planning for the generic model, but the result is an end product of much higher quality. If I had to do it over, I would probably make the same choices and not sacrifice quality because of deadlines unless absolutely necessary. However, I would be able to make a more realistic time planning in advance, and reserve sufficient time for continuous documentation of progress and choices.

10.4 Contribution of this study

The contribution of this study is threefold and has a societal, a methodological and a scientific aspect.

Societal contribution The societal contribution relates to the positioning within the SPREE research project. The thesis proposes a generic agent-based simulation model that can and will be used to inform policy towards absolute decoupling of economic growth from the increasing use of resources on a European level. The model will therefore hopefully contribute to a more sustainable European economy.

Methodological contribution The methodological contribution clearly lies in the proposed additions to, and refinements of, the methodology for developing agent-based models as proposed by Nikolic et al.. The recommendations specifically address required additional techniques to use the methodology for creating a generic model. Also, they provide a set of good modelling practices that will enrich the methodology in general and help future modellers.

Scientific contribution Finally, this study provides a scientific contribution to the field of agent-based modelling, more specifically, the modelling of artificial markets. The model integrates a set of concepts that have not been integrated into one model in previous studies. It proposes a coherent approach to integrate

rational and non-rational considerations and decision making of both producers and consumers, as well as resulting material flows and impacts. Also, the model features sophisticated market research as a novel basis for the decision making of agents in an artificial market. Finally, the generic nature make the model very suitable for further development and to serve as inspiration for future models of artificial markets.

Bibliography

Afman, MR; Chappin, EJL; Jager, W, and Dijkema, GPJ. Agent-based model of transitions in consumer lighting. In *Proceedings of 3rd World Congress on Social Simulation, Kassel, Germany*, 2010.

Axtell, R. Why agents ? on the varied motivations for agent computing in the social sciences. *Technical Report!*, 17, 2000.

Baines, T S; Lightfoot, H W; Evans, S; Neely, A; Greenough, R; Peppard, J; Roy, R; Shehab, E; Braganza, A; Tiwari, A; Alcock, J R; Angus, J P; Bastl, M; Cousens, A; Irving, P; Johnson, M; Kingston, J; Lockett, H; Martinez, V; Michele, P; Tranfield, D; Walton, I M, and Wilson, H. State-of-the-art in product-service systems. *Journal of Engineering Manufacture*, 221(B):1543–1552, 2007.

Behdani, Behzad. Evaluation of paradigms for modeling supply chains as complex socio-technical systems. In *Simulation Conference (WSC), Proceedings of the 2012 Winter*, pages 1–15. IEEE, 2012.

Bernardino, João PR and Araújo, Tanya. On positional consumption and technological innovation: an agent-based model. *Journal of Evolutionary Economics*, pages 1–25, 2013.

Briceno, Tania and Sagel, Sigrid. The role of social processes for sustainable consumption. *Journal of Cleaner Production*, 14(17):1541–1551, 2006.

Chang, Ting-Hua; Lee, Jun-Yen, and Chen, Ru-Hwa. The effects of customer value on loyalty and profits in a dynamic competitive market. *Computational Economics*, 32(3):317–339, 2008.

Chappin, Emile JL; Dijkema, Gerard PJ; van Dam, Koen Haziël, and Lukszo, Zofia. Modeling strategic and operational decision-makingâĂŤan agent-based model of electricity producers. In *The 2007 European Simulation and Modelling Conference, St. Julians, Malta, Eurosis*, 2007.

Choi, J.; Im, N., and Park, J. Agent based model for estimating hybrid electric vehicle market: The case of korea. volume 44, pages 26–33, 2012. URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84879431360&partnerID=40&md5=f30135e6a06d15750cc631967140b1a6`. cited By (since 1996)0.

Choo, Sangho and Mokhtarian, Patricia L. What type of vehicle do people drive? the role of attitude and lifestyle in influencing vehicle type choice. *Transportation Research Part A: Policy and Practice*, 38(3):201 – 222, 2004. ISSN 0965-8564. doi: http://dx.doi.org/10.1016/j.tra.2003.10.005. URL `http://www.sciencedirect.com/science/article/pii/S096585640300096X`.

Chu, Junying; Wang, Can; Chen, Jining, and Wang, Hao. Agent-based residential water use behavior simulation and policy implications: A case-study in beijing city. *Water resources management*, 23(15): 3267–3295, 2009.

Davis, Chris; Nikolic, Igor, and Dijkema, Gerard PJ. Integrating life cycle analysis with agent based modeling: Deciding on bio-electricity. In *Infrastructure Systems and Services: Building Networks for a Brighter Future (INFRA), 2008 First International Conference on*, pages 1–6. IEEE, 2008.

de Haan, Peter; Mueller, Michel G, and Scholz, Roland W. How much do incentives affect car purchase? agent-based microsimulation of consumer choice of new carsâĂŤpart ii: Forecasting effects of feebates based on energy-efficiency. *Energy Policy*, 37(3):1083–1094, 2009.

Deissenberg, C; van der Hoog, S, and Dawid, H. EURACE: A massively parallel agent-based model of the European economy. *Applied Mathematics and Computation*, 204(2):541–552, 2008. doi: 10.1016/j.amc.2008.05.116.

Devisscher, Tahia and Mont, Oksana. An analysis of a product service system in Bolivia: coffee in Yungas. *International Journal of Innovation and Sustainable Development*, 3:262–284, 2008.

Edmonds, Bruce and Moss, Scott. *From KISS to KIDS–an âĂŸanti-simplisticâĂŹmodelling approach*. Springer, 2005.

Eppstein, Margaret J; Grover, David K; Marshall, Jeffrey S, and Rizzo, Donna M. An agent-based model to study market penetration of plug-in hybrid electric vehicles. *Energy Policy*, 39(6):3789–3802, 2011.

European Commission, . Sustainable growth - for a resource efficient, greener and more competitive economy, 2012. URL `http://ec.europa.eu/europe2020/europe-2020-in-a-nutshell/priorities/sustainable-growth/index_en.htm`.

European Environment Agency, . The European Environment: State and Outlook 2010, 2010.

European Environment Agency, . Material resources and waste âĂŤ 2012 update. Technical report, Copenhagen, 2012.

Eurostat, . *Sustainable development in the European Union*. Office for Official Publications of the European Communities, Luxembourg, 2011. ISBN 978-92-79-18516-8. doi: 10.2785/1538.

Evans, David and Jackson, Tim. Towards a sociology of sustainable lifestyles. 2007.

Fishbein, Bette K; McGarry, Lorraine S, and Dillon, Patricia S. Leasing: A Step Toward Producer Responsibility. Technical report, New York, USA, 2000.

Flint, Mary Louise. *IPM in practice: principles and methods of integrated pest management*, volume 3418. UCANR Publications, 2012.

Givoni, Moshe; Macmillen, James; Banister, David, and Feitelson, Eran. Strategic policy packaging in complex domains. 2011.

He, Hong-Wei and Balmer, John MT. A grounded theory of the corporate identity and corporate strategy dynamic: A corporate marketing perspective. *European Journal of Marketing*, 47(3/4):401–430, 2013.

Hirschl, B; Konrad, W, and Scholl, G. New concepts in product use for sustainable consumption. . *Journal of Cleaner Production*, 11:873–881, 2003.

Hommes, C H. Chapter 23 Heterogeneous Agent Models in Economics and Finance, 2006. URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-66049139912&partnerID=40&md5=bb6b277af30d6fe54295b37f103802ac`.

Itoh, Satoshi; Murakami, Yasuyuki, and Iba, Takashi. Consumer network and market dynamics. In *JCIS*, 2006.

Janssen, Marco A and Jager, Wander. Stimulating diffusion of green products. *Journal of Evolutionary Economics*, 12(3):283–306, 2002.

Jordan, Patrick W. Human factors for pleasure in product use. *Applied Ergonomics*, 29(1):25 – 33, 1998. ISSN 0003-6870. doi: http://dx.doi.org/10.1016/S0003-6870(97)00022-7. URL `http://www.sciencedirect.com/science/article/pii/S0003687097000227`.

Karacapilidis, Nikos and Moraïtis, Pavlos. Intelligent agents for an artificial market system. In *Proceedings of the fifth international conference on Autonomous agents*, pages 592–599. ACM, 2001.

Katagiri, Hideki; Nishizaki, Ichiro; Hayashida, Tomohiro, and Daimaru, Takahiro. An agent-based simulation model for analysis on marketing strategy considering promotion activities. In *Agent and Multi-Agent Systems: Technologies and Applications*, pages 619–628. Springer, 2009.

Kiesling, Elmar; Günther, Markus; Stummer, Christian, and Wakolbinger, Lea M. Agent-based simulation of innovation diffusion: a review. *Central European Journal of Operations Research*, 20(2):183–230, 2012.

Kirman, Alan. Artificial markets: Rationality and organisation. In *Complexity and Artificial Markets*, pages 195–234. Springer, 2008.

Kotakorpi, Elli; Lähteenoja, Satu, and Lettenmeier, Michael. Household MIPS: Natural resource consumption of Finnish households and its reduction. Technical report, Helsinki, 2008.

Laciana, Carlos E and Oteiza-Aguirre, Nicolás. An agent based multi-optional model for the diffusion of innovations. *Physica A: Statistical Mechanics and its Applications*, 2013.

Liu, Hongliang; Howley, Enda, and Duggan, Jim. The impact of market preferences on the evolution of market price and product quality. In *MALLOW*, 2010.

Lutter, S; Polzin, C; Giljium, S; PÃąlfy, T; Patz, T; Dittrich, M; Kernegger, L, and Rodrigo, A. Under pressure: How our material consumption threatens the planetÂťs water resources. Technical report, Vienna, 2009.

Maidstone, Robert. Discrete event simulation, system dynamics and agent based simulation: Discussion and comparison. *System*, pages 1–6, 2012.

Maya Sopha, Bertha; Klöckner, Christian A, and Hertwich, Edgar G. Exploring policy options for a transition to sustainable heating system diffusion using an agent-based simulation. *Energy Policy*, 39(5):2722–2729, 2011.

Midgley, D; Marks, R, and Kunchamwar, D. The building and assurance of agent-based models: an example and challenge to the field. *Journal of Business Research*, 60:884–893, 2007.

Mont, Oksana. Product-Service Systems: Shifting corporate focus from selling products to selling product-services: a new approach to sustainable development. Technical report, 2000.

Mont, Oksana. *Product-service systems: Panacea or myth?* PhD thesis, Lund, Sweden, 2004.

Mont, Oksana and Lindhqvist, T. The role of public policy in advancement of product service systems. *Journal of Cleaner Production*, 11:905–914, 2002.

Mueller, Michel G and de Haan, Peter. How much do incentives affect car purchase? agent-based microsimulation of consumer choice of new carsâĂŤpart i: Model structure, simulation of bounded rationality, and model validation. *Energy Policy*, 37(3):1072–1082, 2009.

Neri, Filippo. Using an agent based simulation to evaluate scenarios in customers' buying behaviour. In *Emergent Intelligence of Networked Agents*, pages 177–188. Springer, 2007.

Ng, Desmond. Understanding the market dynamics of entrepreneurial networks. *Journal on Chain and Network Science*, 8(2):93–105, 2008.

Nikolic, Igor; Van Dam, Koen H, and Lukszo, Zofia. *Agent-based modelling of socio-technical systems.* Springer, 2012.

Ogibayashi, Shigeaki and Takashima, Kousei. Multi-agent simulation of fund circulation in an artificial economic system involving self-adjusted mechanism of price, production and investment. In *Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on*, pages 1127–1130. IEEE, 2009.

Ohori, Kotaro and Takahashi, Shingo. Market design for standardization problems with agent-based social simulation. *Journal of Evolutionary Economics*, 22(1):49–77, 2012.

Rothenberg, S. Sustainability through Servicizing. *MIT Sloan Management Review*, 48(2):83–91, 2007.

Said, Lamjed Ben; Bouron, Thierry, and Drogoul, Alexis. Agent-based interaction analysis of consumer behavior. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 184–190. ACM, 2002.

Sasaki, Takao; Becker, D Vaughn; Janssen, Marco A, and Neel, Rebecca. Does greater product information actually inform consumer decisions? the relationship between product information quantity and diversity of consumer decisions. *Journal of economic psychology*, 32(3):391–398, 2011.

Schmidt-Bleek, F and Lehner, F. *The MIPS-Concept. Fewer consumption of natural resources - more quality of life through Factor 10.* Droemer, Munich, 1998.

Schwartz, Barry; Ward, Andrew; Monterosso, John; Lyubomirsky, Sonja; White, Katherine, and Lehman, Darrin R. Maximizing versus satisficing: happiness is a matter of choice. *Journal of personality and social psychology*, 83(5):1178, 2002.

Schwarz, Nina and Ernst, Andreas. Agent-based modeling of the diffusion of environmental innovationsâĂŤan empirical approach. *Technological forecasting and social change*, 76(4):497–511, 2009.

Shalizi, Cosma Rohilla. Methods and techniques of complex systems science: An overview. In Deisboeck, T.S. and Kresh, J. Y., editors, *Complex systems science in biomedicine.*, chapter 1, pages 33–114. Springer, New York, 2006.

Silver, Steven D. *Status Through Consumption: Dynamics of Consuming in Structured Environments*. Kluwer Academic Publishers, Norwell, MA, 2002.

Sperling, D; Shaheen, S, and Wagner, C. Carsharing - Niche market or new pathway?, 2000.

SPREE, . About the project, 2012. URL `http://www.spreeproject.com/?page_id=6`. Accecssed: 14 October 2013.

Tay, Nicholas SP and Lusch, Robert F. Agent-based modeling of ambidextrous organizations: Virtualizing competitive strategy. *Intelligent Systems, IEEE*, 22(5):50–57, 2007.

Terano, Takao and Naitoh, Kenichi. Agent-based modeling for competing firms: from balanced-scorecards to multiobjective strategies. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, pages 8–pp. IEEE, 2004.

Tesfatsion, L. Agent-based computational economics: growing economies from the bottom up. *Artificial life*, 8(1):55–82, 2002. URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-0036359150&partnerID=40&md5=a8823f8731c77ca78a648e3e44c6b462`.

Tesfatsion, L. Chapter 16 Agent-Based Computational Economics: A Constructive Approach to Economic Theory, 2006. URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-66049110085&partnerID=40&md5=aa199431d2529a3bd2410faec0fb7216`.

Tesfatsion, L. SPREE Partners, 2013. URL `http://www.spreeproject.com/?page_id=10`. Accessed: 18 November 2013.

Toffel, Michael W. Contracting for Servicizing. 2002.

Tukker, Arnold and Jansen, Bart. Environmental Impacts of Products. *Journal of Industrial Ecology*, 10(3): 159–182, 2006.

Weaver, P; Jansen, L; Van Grootveld, G; Van Spiegel, E, and Vergragt, P. *Sustainable Technology Development*. Greenleaf Publishing Ltd., Sheffield, 2000.

White, Allen L; Stoughton, Mark, and Feng, Linda. Servicizing: The Quiet Transition to Extended Product Responsibility. Technical report, 1999.

Zenobia, Brent; Weber, Charles, and Daim, Tugrul. Artificial markets: A review and assessment of a new venue for innovation research. *Technovation*, 29(5):338–350, 2009.

Zhang, Tao and Zhang, David. Agent-based simulation of consumer purchase decision-making and the decoy effect. *Journal of Business Research*, 60(8):912–922, 2007.

A | SPREE factsheet

SPREE PROJECT

SPREE PROJECT AT A GLANCE

Title: Servicizing Policy for Resource Efficient Economy

Instrument: Seventh Framework Programme (FP7)

Total Cost: 3,011,479.00 €

EC Contribution: 2,399,663.20 €

Duration: 36 months

Start Date: 1.7.2012

Project Coordinator: Prof. Eugenijus Butkus, Research Council of Lithuania

Project Manager: Ms. Yael Marom, Jerusalem Institute for Israel Studies

SPREE CONSORTIUM
Research Council of Lithuania
School for Geography and Environment, Transport Studies Unit, Oxford University
Centre for Environmental Strategy, Surrey University
The International Institute for Industrial Environmental Economics, Lund University
ICEDE Research Group, University of Santiago de Compostel
The Finnish Environment Institute
Faculty of Technology, Policy and Management, Delft University of Technology
The Jerusalem Institute for Israel Studies
The Faculty of Management, Tel Aviv University
The department of Business Management, Ben Gurion University of the Negev

THE CHALLENGE

Green growth policies, improving resource productivity through supply side measures such as supporting eco-innovation or facilitating sustainable use of raw materials, have achieved only relative decoupling since they have inevitably led to rebound effects through the increased scale of consumption. The challenge is to design policy instruments which integrate both supply and demand side policies influencing both production and consumption patterns together with positive societal effects. SPREE Project is constructed with the overarching goal to assist and enhance the development of a resource-efficient Europe, characterized by an absolute decoupling of economic growth and social prosperity from unsustainable use of resources.

PROJECT OBJECTIVES

SPREE project proposes to face this challenge through Servicizing policy directed at achieving absolute decoupling together with positive societal advantages. The objectives of SPREE project are: (1) to understand the basic components of Servicizing systems; (2) to develop tools to explore the economic, social and environmental impacts of restructuring the economy towards servicizing economy; (3) to evaluate Servicizing systems, using three sector-specific Agent Based Models in the water, mobility and agri-food sectors, within the partners' local countries conditions; (4) to integrate societal objectives into the environmental agenda; (5) to test the effect of different Servicizing policies; (6) to provide a set of methods for pursuing and evaluating policies and to consider a set of scenarios in which different policy mixes are implemented, as a way of offering guidelines to progress towards Spree and; (7) to translate the knowledge gained throughout the project to tangible resources - 'Servicizing Policy Packages'.

METHODOLOGY

SPREE project will use existing tools such as Life Cycle Assessment (LCA), Input-output Analysis (I/OA) and hybrid I/O-LCA, and develop new tools, e.g. dynamic LCA, to capture the sustainability of the fundamentally different production structure and value chains of Servicizing systems. In addition, in order to measure absolute decoupling and social impacts, the project will use novel Agent Based Modelling (ABM) stimulating how Servicizing systems emerge and test outcomes of proposed policies and their effect on achieving absolute decoupling. The social aspects will be evaluated through the development of social metrics related to Servicizing impacts (for example, equity access to opportunities). Qualitative methodologies will be used such as household surveys and 'Triple Task' participatory methodology bringing together key actors and policymakers. Finally, Policy Packages methodology will allow ex-ante identification of possible contradictions among proposed policy measures and synergies between such measures, thereby producing packages where total contribution exceeds the sum of its parts.

EXPECTED RESULTS

The key outcome of SPREE project is 'Servicizing Policy Packages' to achieve a truly sustainable EU economy and assess the contribution of policies to decoupling, mitigation of rebound effects and social desired outcomes. In addition, SPREE project will provide: (1) a thorough understanding of the transition dynamics of Servicizing systems from the current conditions into a truly Servicizing economy; (2) a quantitative tool (ABM) for testing adaptive policies directed at promoting the transition to Servicizing opportunities; (3) a bottom up effect explorations of policies; (4) the development and use of new ways to measuring and visualizing decoupling success through the modelling simulation outcomes; (5) a better understanding of the existing differences within the European Union and the extended region countries with regards to customer approach, production standards, overall advancement on environmental issues and available infrastructure and; (6) contribution to methodology and research development coupled with support to actual policymaking.

Visit us at:

SPREE website: www.spreeproject.com

SPREE facebook page: http://www.facebook.com/SpreeProject

B Elements and structure

B.1 Introduction

The previous chapter has presented the main interactions between agents and objects in the SPREE generic model. This chapter discusses the corresponding ontology, that results from the system decomposition and concept formalisation steps (2 and 3, respectively) of the methodology. The ontology identifies the model elements and formalises relations between them. It also provides the states and behaviour of the model elements in a formalised way that is directly translatable to computer language.

Section B.2 first presents the list of model agents and objects. Section B.3 provides object interrelation diagrams that visualise how the agents and objects are integrated into one model. Sections B.4, B.5, B.6 and B.7 present the states and behaviours of the Observer, agents, objects and links, respectively. Section B.8 lists some of the additional assumptions and limitations that were introduced during the conceptualisation process.

B.2 List of agents and objects

Box B.1 provides a list of the elements that the SPREE generic model consists of. It completes the system identification step (step 2 of the methodology). The figure distinguishes the Observer, agents, objects and links as the four high-level categories of elements. Agents represent individuals, organisations, the 'world market' and the 'physical environment'. Objects represent 'things'. Box B.1 further makes the distinction between active and passive agents, as well as tangible and intangible objects. Active agents and tangible objects can interact with each other. Passive agents and intangible objects basically just contain information that agents and objects can use. The figure contains a brief description of each model element. Later sections in this chapter provide much more detail on each of the elements.

Observer	
Observer (O)	Does not participate in the market, but can influence it as external 'force'
Agents	
Active market participants	
Producing Business (PB)	Produces and sells Products and Services
Consuming Business (CB)	Consumes Products/Services to satisfy Need
Consuming Individual (CI)	Consumes Products/Services to satisfy Need
Passive agents	
World Market (WM)	Sells and buys Products
Physical Environment (PE)	Accepts wastes
Objects	
Tangible objects (can be `owned' by agents)	
Tool (Product) (TP)	Remaining usability measured in use time
Consumable (Product) (CP)	Remaining usability measured in quantity
Manufacturing Model (MM)	Basis for PB production
Sales Model (SM)	Results in offered Product or Service
Consumption Model (CM)	Specifies Product/Service consumption
Intangible objects (just contain information)	
Service (S)	Defines a service produced by an SM
Infrastructure (I)	Specifies a relevant infrastructure
Skill (SK)	Specifies a relevant skill
Resource (R)	Represents a resource such as steel or CO_2
Policy Package (PP)	Collection of Policy Instruments
Policy Instrument (PI)	Collection of Policy Effects
Policy Instrument Effect (PIE)	Defines a specific effect on the market
Market Development Event (MD)	Collection of Market Development Effects
Market Development Effect (MDE)	Defines a specific effect on the market
Links	
Service Contract	Connects seller and buyer of a Service

Box B.1: List of elements of the SPREE generic model

B.3 Object interrelation diagrams

The list in box B.1 is still just a list of elements. An ontology must also relate the model elements together. This section provides three diagrams that specify the type of interactions between the observer, agents, objects and links.

B.3.1 Agent interactions

Figure B.2 illustrates the primary flows between agents, as well as the Observer's position 'on top of' the market. The Observer influences the market by (de)activating Policy Instruments (structured in Packages) and Market Development Events. The figure again makes the distinction between active and passive agents.

Active agents can buy from or sell to the World Market, and dump wastes in the Physical Environment. PBs sell Products and Services to the two types of CAs. They perform market research among the CAs that helps them in their strategic and tactical decisions.

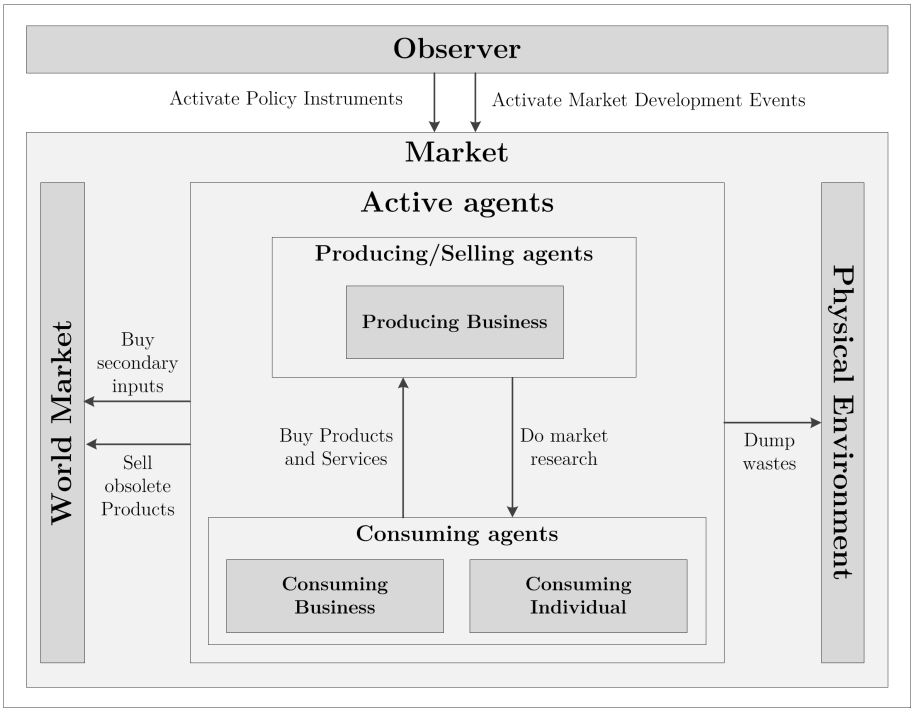

Figure B.2: Relations between agents in the model

B.3.2 Material flows

Figure B.3 only depicts model *objects*. It illustrates that all material flows originate from the World Market, are processed by Manufacturing Models (MMs), Sales Models (SMs) and/or Consumption Models (CMs) and then end up in the Physical Environment (PE) or are sold to the World Market (WM). Stocks of (material) main inputs and outputs are located at SMs and CMs. The distinction between product-based and service-based SMs allow partial and gradual servicising, where a PB can gradually allocate more production capacity to either of the SMs.

B.3.3 Agent-object relations

Figure B.4 is more complex as it includes all model elements in one diagram. It specifies connections between agents and objects. Producing Businesses (PBs) choose an MM and up to two SMs to be able to sell Products

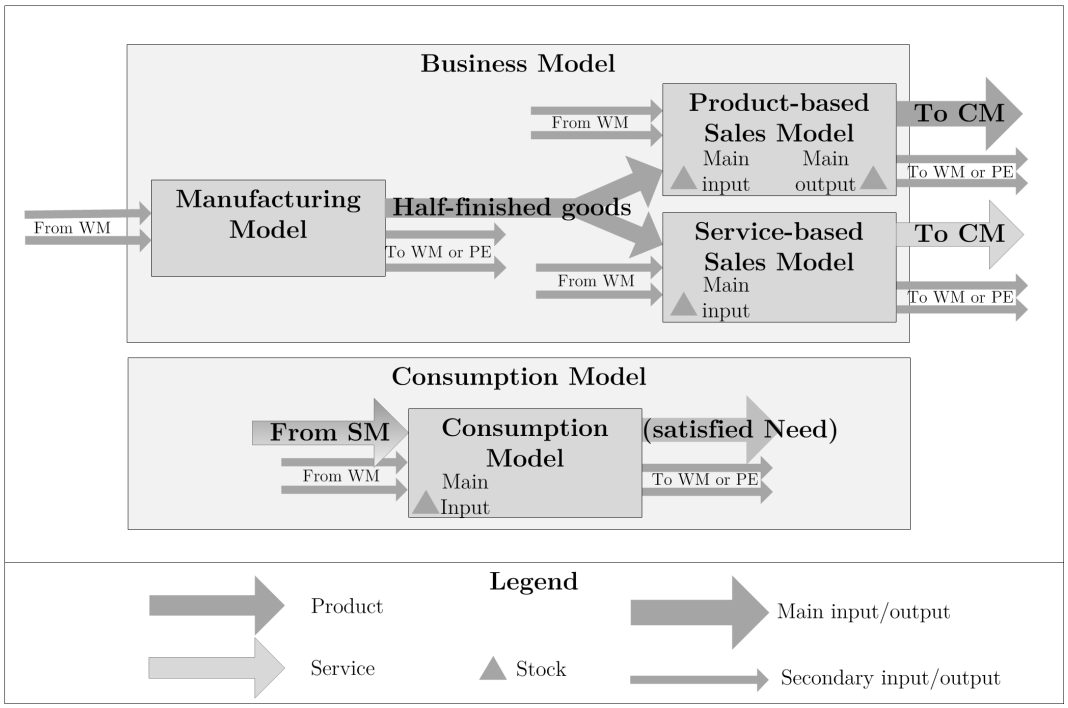

Figure B.3: Material flows in the model

and Services to consuming agents (CAs), which in turn have a CM to satisfy their Need. Agents can connect to Infrastructures and learn Skills that both function as prerequisites for CMs, MMs and SMs. Products and Services originate from the World Market, and can be either sold to the World Market or dumped in the Physical Environment. Products and Services have associated material extraction and generated waste in the form of a set of Resources. Service Contracts are links that represent an agreement between a PB and a CA on the delivery of a Service over a longer time span (for esthetic reasons, the figure does not explicitly relate Service Contracts to CAs and PBs). Finally, the Observer can (de)activate Policy Instruments (structured in Packages) and Market Development Events, the Effects of which can influence any property of agents and objects in the model.

B.4 The Observer

The previous two sections have identified the list of model elements and the relations between them. This and the following sections expand on the observer, the agents, the objects and the links in the model by identifying their states (or properties) and behaviours. States refer to agent properties at a certain point in time that may or may not change as a result of model interaction. Behaviours are things that the

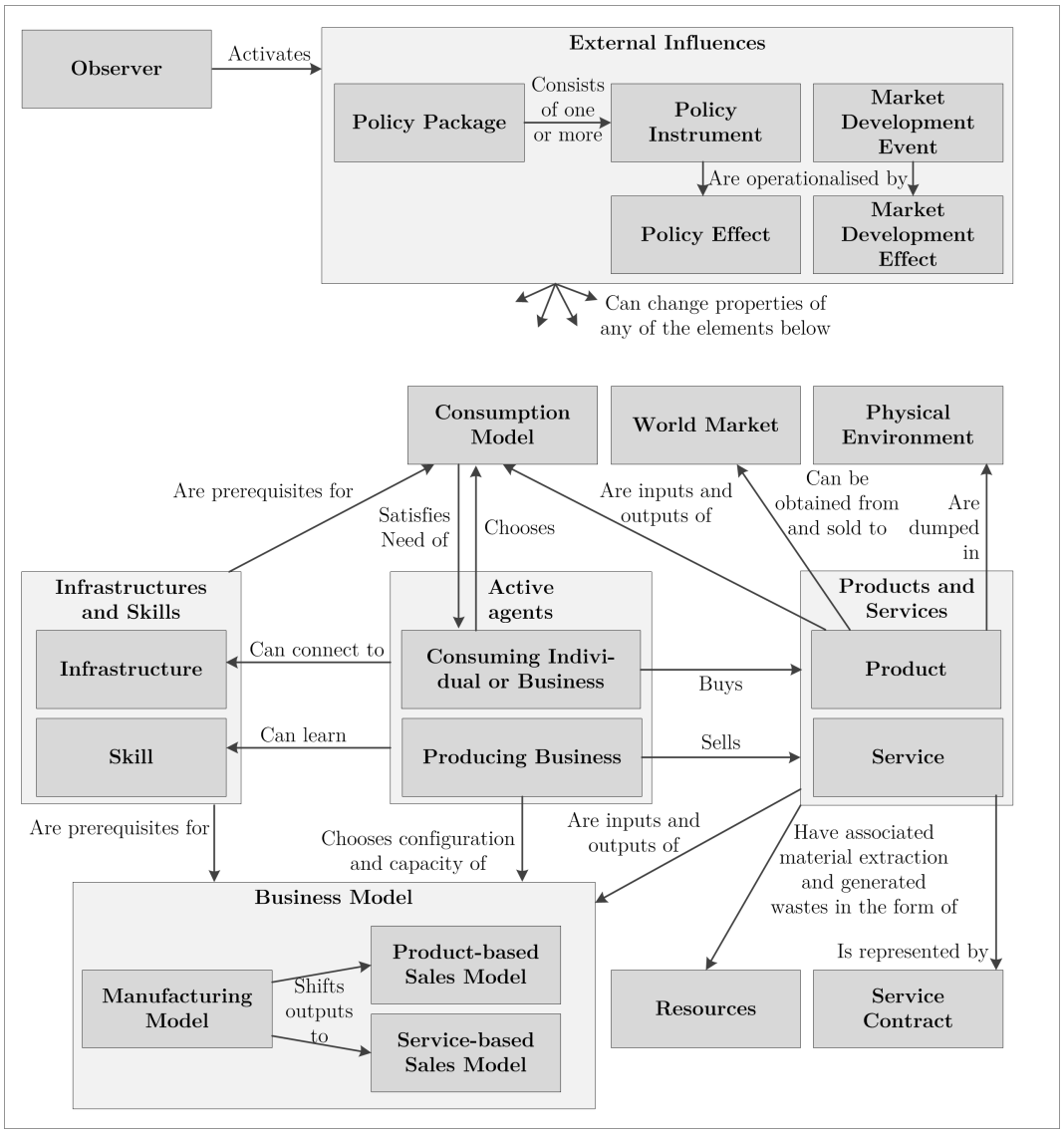

Figure B.4: All model elements in one diagram

model element can do in the model, such as making decisions, buying products or learning a Skill. The implemented model has a substantial number of technical support variables that are not included in these tables. Chapter 9 will elaborate on implementation details, including the role of 'the library agents/objects' that are mentioned in some of the tables. For now, it suffices to think of library agents/objects as a 'box' around a group of model elements with a shared group ID.

The Observer (box B.5) has a special status within the model and can be interpreted as the model 'controller'. The Observer controls global variables such as the basic time unit (btu) and currency. The

global variables also include the sets of Skills, Infrastructures and Resources that are relevant for the specified market, as well as the Policy Packages and Market Development Events that may affect the market.

The Observer can actively influence the market through Policy Instruments (structured in Packages) and Market Development Events. The Observer also collects aggregate data and statistics.

Observer ('controller')	
Global variables	
State	*Specified as*
Basic time unit (btu)	String
Currency	String
Library Agents	Set of agent library instances
Library Objects	Set of object library instances
Product and Service meta-properties	
ID	List of Integers
Name	List of Strings
Relevant Resources	
ID	List of Integers
Name	List of Strings
Infrastructures	Infrastructure agentset
Skills	Skill agentset
Policy Packages	Policy Package agentset
Market Development Events	Market Development Event ag.set
Behaviour/rules	
Activate/deactivate Policy Instruments	
Activate/deactivate Market Developments	
Calculate market shares	
Calculate total use of resources	
Calculate total associated wastes	
Calculate other aggregate statistics	

Box B.5: The Observer

B.5 Agents

B.5.1 Active agents

The model has three types of active agents: Producing Businesses, Consuming Businesses (CBs) and Consuming Individuals (CIs). They all have a cash balance, certain preferences (in the form of thresholds and weights) representing their strategy or lifestyle, and a willingness to pay for the strategic or lifestyle fit. All three agent types also have specified intervals between strategic and tactical reconsiderations, as well as a return on investment period that governs how far they look ahead in time. All active agents may have signed Service Contracts. Furthermore, the agents can learn Skills and connect to Infrastructures (with a difference between *available* Infrastructures and Infrastructures that the agent is actually *connected* to.

Producing Businesses The three agent types have some unique additional states and behaviour. Producing Businesses (see box B.6) have a 'capacity over expected sales ratio' that specifies their desired buffer capacity relative to expected sales. The market research fraction specifies the fraction of CAs that the PB 'consults' when doing market research. The market research risk factor makes estimates of expected sales more conservative, taking into account the dynamic nature of a competitive market. This state takes into account, for instance, that competitors will respond to this PB's choices, possibly leading to lower sales than expected. Furthermore, PBs have an MM and up to two SMs: one product-based (SM-P) and one service-based (SM-S).

PBs can evaluate alternative business models (MM + SM-P + SM-S), adopt new MMs and SMs and change their production capacity level. They can optimise the selling prices of their main outputs and adjust target stock levels accordingly. PBs perform market research to support their strategic and tactical decisions. On the operational level, PBs produce Products and deliver Services. To this end, they buy inputs from the World Market and dump secondary outputs in the Physical Environment or sell them to the World Market. They sell the Products and Services to CAs.

Consuming Agents There are two types of CAs: Consuming Businesses (CBs, see box B.7) and Consuming Individuals (CIs, see box B.8). CAs have a certain Need per btu, which is their primary reason to participate on the artificial market. They keep track of their current supplier, i.e. the PB that they have last bought something from. An additional preference-related attribute indicates how much they are willing to pay to stay with a current supplier (thus representing loyalty).

Consuming Businesses also have 'default' costs and revenues. For instance, a grower of tomatoes may have to make *default costs*, other than those relating to pest management, of 800,000 euro. She has a Need for pest management functionality and thus participates in the 'pest management' market. Her default method of pest management, i.e., buying her own machinery and operating them herself, leads to additional costs of 50,000 euro per year (depending, of course, on actual market prices) and expected *default revenues* (from selling tomatoes) of 1,000,000 euro per year. The additional costs, as well as the revenues, may differ if the PB makes different choices (see also box B.15 later in this chapter).

B.5.2 Passive agents

The model has two types of passive agents: the World Market (box B.9) and the Physical Environment (figure B.10). The World Market contains buying and selling prices of all relevant Products (not Services). The Physical Environment contains dump 'prices' (or costs), such as costs associated with CO_2 emission rights. Neither of these agents have active behaviour.

Producing Business	
States	
State	*Specified as*
Library id	Integer
Cash balance	Real
Strategic preferences	
Thresholds	List of Integers 1...10
Weights	List of Integers 0...5
Willingness to pay for strategic fit	Real 0...1 (relative to best profit)
Decision-making	
Capacity over expected sales ratio	Real
Market research fraction	Real 0...1
Market research risk factor	Real 0...1
Return on investment period	Integer (in btu's)
Strategic reconsideration interval	Integer (in btu's)
Tactical reconsideration interval	Integer (in btu's)
Business Model	
Manufacturing Model	MM instance
Product-based Sales Model	SM instance
Service-based Sales Model	SM instance
Active Service Contracts	Service Contract instances
Available infrastructures	List of Integers (Infr. library id's)
Connected infrastructures	List of Integers (Infr. library id's)
Acquired skills	List of Integers (Skill id's)
Behaviour/rules	
Evaluate alternative business models	
Adopt new MM/ SM/ capacity	
Optimise selling price and target stock level	
Perform market research to obtain consumer demand	
Produce Products / Deliver Services	
Buy from the World Market	
Sell obsolete stocks and reusable outputs to the World Market	
Dump in the Physical Environment	
Sell Products/Services	
Learn Skills	
Connect to Infrastructures	

Box B.6: Producing Business

B.6 Objects

B.6.1 Tangible objects

The SPREE generic model includes five types of tangible objects, which represent subtypes of Products and transformation models.

Product subtypes Tools (box B.11) and Consumables (box B.12) are both subtypes of Products. Products can be created in two ways: by buying them from the World Market or by creating them (from other

Consuming Business	
States	
State	*Specified as*
Library id	Integer
Cash balance	Real
Default revenue per btu	Real
Default cost per btu	Real
Need per btu	Real
Strategic preferences	
Thresholds	List of Integers 1...10
Weights	List of Integers 0...5
Willingness to pay for strategic fit	Real 0...1 (relative to best profit)
Willingness to pay for loyalty	Real 0...1 (relative to best profit)
Decision-making	
Return on investment period	Integer (in btu's)
Strategic reconsideration interval	Integer (in btu's)
Tactical reconsideration interval	Integer (in btu's)
Consumption Model	
Current Supplier	PB instance
Active Service Contract	Service Contract instance
Available infrastructures	List of Integers (Infr. library id's)
Connected infrastructures	List of Integers (Infr. library id's)
Acquired skills	List of Integers (Skill id's)
Behaviour/rules	
Evaluate alternative Consumption Models	
Adopt new CM	
Find cheapest (with loyalty correction) supplier for current CM	
Satisfy Need	
Buy Products/Services from PBs	
Buy from the World Market	
Dump in the Physical Environment	
Learn Skills	
Connect to Infrastructures	

Box B.7: Consuming Business

inputs) by using MM and SM processes. Products on the World Market have a pre-specified use of Resources. Products created by MMs and/or SMs have associated resource use and waste generation that are calculated based on the resource use and associated waste of all corresponding inputs and secondary outputs. They are always owned by an Agent. The only difference between Tools and Consumables is that consumption of a Tool consumes part of its remaining *use time*, whereas consumption of a Consumable decreases the remaining *quantity*.

Transformation model subtypes Products and Services that cannot be obtained from the World Market are produced through MMs and SMs, then sold on the market and converted into consumer Needs by CMs. This makes MMs, SMs and CMs all transformation models. They share a number of 'state' aspects (or

Consuming Individual	
States	
State	*Specified as*
Library id	Integer
Cash balance	Real
Need per btu	Real
Lifestyle-based preferences	
Thresholds	List of Integers 1...10
Weights	List of Integers 0...5
Willingness to pay for strategic fit	Real 0...1 (relative to best profit)
Willingness to pay for loyalty	Real 0...1 (relative to best profit)
Decision-making	
Return on investment period	Integer (in btu's)
Strategic reconsideration interval	Integer (in btu's)
Tactical reconsideration interval	Integer (in btu's)
Consumption Model	
Current Supplier	PB instance
Active Service Contract	Service Contract instance
Available infrastructures	List of Integers (Infr. library id's)
Connected infrastructures	List of Integers (Infr. library id's)
Acquired skills	List of Integers (Skill id's)
Behaviour/rules	
Evaluate alternative Consumption Models	
Adopt new CM	
Find cheapest (with loyalty correction) supplier for current CM	
Satisfy Need	
Buy Products/Services from PBs	
Buy from the World Market	
Dump in the Physical Environment	
Learn Skills	
Connect to Infrastructures	

Box B.8: Consuming Individual

World Market	
Properties	
State	*Specified as*
Selling prices	
Product IDs	List of Integers
Corresponding prices	List of Reals
Buying prices	
Product IDs	List of Integers
Corresponding prices	List of Reals

Box B.9: World Market

simply states) but also have unique states.

All transformation models have an 'availability' state that indicates whether agents are allowed and able to choose them. The main reasons for unavailability include policy-related bans and innovation (they may

Physical Environment	
Properties	
State	*Specified as*
Dump prices	
Product IDs	List of Integers
Corresponding dump prices	List of Reals

Box B.10: Physical Environment

Tool (Product)	
States	
State	*Specified as*
Library id	Integer
Name	String
Owner	Agent instance
Remaining use time	Real
Associated use of resources	List of Reals
Associated wastes	List of Reals

Box B.11: Tool

Consumable (Product)	
States	
State	*Specified as*
Library id	Integer
Name	String
Owner	Agent instance
Remaining quantity	Real
Associated use of resources	List of Reals
Associated wastes	List of Reals

Box B.12: Consumable

not yet have been 'invented'). Transformation models further have preconditions in the form of Skills and Infrastructures. They have associated costs in three categories: initial investment costs represent the costs of required permanent physical and organisational changes. Structural costs represent the costs of production facilities over the long run, including depreciation and maintenance. Variable costs per unit relate to variable costs *other than* the costs of explicitly included inputs and wastes. Costs of labour and electricity are good examples.

All transformation models also specify the types and amounts of inputs and secondary outputs per unit

Manufacturing Model	
States	
State	*Specified as*
Library id	Integer
Available?	Boolean
Preconditions	
Required Skill IDs	List of Integers
Required Infrastructure IDs	List of Integers
Costs	
Initial investment cost	Real
Structural costs per time unit per unit of capacity (representing depreciation, maintenance, etc.)	Real
Variable costs per unit of output	Real
Input/output transformation	Integer
Main output ID	
'Secondary' inputs	List of Integers
IDs	List of (corresponding) Reals
Amounts per unit of main output	
Secondary outputs	List of Integers
IDs	List of (corresponding) Reals
Amounts per unit of main output	Real
Production capacity per btu	

Box B.13: Manufacturing Model

of main output (which is a unit of the Need in the case of CMs). The main output of an MM is the main input of associated SMs. The main outputs of SMs are then the Products and Services that can be sold and function as main inputs of CMs. Secondary inputs are always obtained from the World Market. Secondary outputs are either sold to the World Market or dumped in the Physical Environment.

MMs (box B.13) have a unique property relating to the current production capacity. The owning PB can change the capacity as a part of its strategic reconsideration process.

SMs (box B.14) and CMs (box B.15) keep input 'stocks' (representing associated owned Products in the case of product-based CMs). Product-based SMs also have output stocks, ready to be sold to the market. SMs and CMs also both have associated 'scores' on the aspects of PB and CA preferences. These are the scores used during the decision-making process of those active agents.

SMs are always linked to a specific type of MM, specified by an appropriate state. Finally, the SM contains price details of its main output, involving three types of prices: one-time contract costs, price per (functional) unit sold and price per time unit. The PB can only change the price per unit in the model, but the other two aspects can still be specified by the user.

CMs have a 'revenue multiplier' that can modifies the revenues of a CB if she uses that CM. This can

Sales Model	
States	
State	*Specified as*
Library id	Integer
Available?	Boolean
Associated MM ID	Integer
Preference scores	List of Integers 1...10
Preconditions	
Required Skill IDs	List of Integers
Required Infrastructure IDs	List of Integers
Costs	
Initial investment cost	Real
Structural costs per time unit (representing depreciation, maintenance, etc.)	Real
Variable costs per unit of output	Real
Input/output transformation	
Main output	
ID	Integer
Use time multiplier (if service)	Real
Main input	
ID	Integer
Amount per unit of main output	List of (corresponding) Reals
Secondary inputs	
IDs	List of Integers
Amounts per unit of main output	List of (corresponding) Reals
Secondary outputs	
IDs	List of Integers
Amounts per unit of main output	List of (corresponding) Reals
Stocks	
Main output stock	Set of produced Products
Main input stock	Set of Products (produced by MM)
Stock/demand ratio (if service)	Real
Offer	
One-time contract cost	Real
Price per unit	Real
Price per btu	Real

Box B.14: Sales Model

be used, for instance, to incorporate that specialised integrated pest management services can lead to better yields and thus higher profits.

B.6.2 Intangible objects

Services Services (box B.16) are sellable outputs of SMs, just like Products. However, Services are not tangible or storable and are directly consumed as they are produced. They are measured in functional units corresponding to their name. Services have associated use of resources and generated wastes that are calculated dynamically as described for Products in section B.6.1. Service Contracts (see below) specify

Consumption Model	
States	
State	*Specified as*
Library id	Integer
Available?	Boolean
Preference scores	List of Integers 1...10
Revenue multiplier	Real
Automatic repurchase?	Boolean
Preconditions	
Required Skill IDs	List of Integers
Required Infrastructure IDs	List of Integers
Costs	
Initial investment cost	Real
Structural costs per time unit (representing depreciation, maintenance, etc.)	Real
Variable costs per unit of output	Real
Input/output transformation	
Main input	
ID	Integer
Amount per unit of Need	Real
Secondary inputs	
IDs	Integer
Amounts per unit of Need	List of (corresponding) Reals
Secondary outputs	
IDs	List of Integers
Amounts per unit of Need	List of (corresponding) Reals
Associated owned Products	Set of Products

Box B.15: Consumption Model

ongoing relations between buyers and sellers of a Service.

Service	
Properties	
Property	*Specified as*
Library ID	Integer
Name	String
Associated resource use	List of Reals
Associated wastes	List of Reals

Box B.16: Service

Infrastructures and Skills Infrastructures (box B.17) and Skills (box B.18) represent preconditions for transformation models. The user specifies which Infrastructures are available to which agents. Agents that have the infrastructure available can connect to it by paying a certain fee per time unit. Similarly, Skills are distributed during initialisation, but agents can learn other Skills when needed, taking into account the Skill

learning costs in their decision-making process.

Infrastructure	
Properties	
Property	*Specified as*
Library ID	Integer
Name	String
Connection fee per time unit	Real

Box B.17: Infrastructure

Skill	
Properties	
Property	*Specified as*
Library ID	Integer
Name	String
Learning costs	Real

Box B.18: Skill

Resources The Resource element purely serves to allow the model to express resource use and generated wastes in some shared format. The Observer contains a fixed set of Resources that are relevant to the specific market that the model represents. For example, the set of Resources may consist of steel, plastic, oil and CO_2. All Products and Services have a property that specifies associated resource use and generate wastes for each Resource type in the set.

Resource	
Properties	
Property	*Specified as*
Library ID	Integer
Name	String

Box B.19: Resource

Policy Policy Packages are conceptualised as three-level structures: Policy Packages consist of Policy Instruments which in turn have specific Policy Effects. Policy Packages (box B.20) also specify the (de)activation conditions for each of the Policy Instruments.

Policy Package	
Properties	
Property	*Specified as*
Library ID	Integer
Activation condition(s)	Logical expression
Deactivation condition(s)	Logical expression
Policy Instruments	Policy Instrument agentset

Box B.20: Policy Package

Similarly, Policy Instruments (box B.21) specify the (de)activation conditions for the associated Policy Effects.

Policy Instrument	
Properties	
Property	*Specified as*
Library ID	Integer
Activation condition(s)	Logical expression
Deactivation condition(s)	Logical expression
Policy Effects	Policy Effect agentset

Box B.21: Policy Instrument

Policy Effects (box B.22) only affect agents and objects that meet a certain logical rule, e.g. ((is PB) OR (is CB)) AND (has CM with (ID = 3)). The Policy Effect specifies the mathematical operation that has to be performed on a certain state of the model elements that meet the selection rule. The mutation can be either temporary (i.e. are reversed when the Policy Instrument is deactivated) or permanent. An example of a temporary effect is a subsidy for certain Services. An example of a permanent effect can be a change of certain agents‘ preferences.

Policy Effect	
Properties	
Property	*Specified as*
Library ID	Integer
Target agent(s)/object(s)	Logical expression
Value mutation	Mathematical operation
Temporary effect?	Boolean

Box B.22: Policy Effect

Market Developments Market Development Events (box B.23) and Market Development Effects (box B.24) have a very similar structure as Policy Instruments and Policy Effects. Conceptually, they represent different things: external influences on the market that can and cannot be controlled by the problem owner(s). However, the only effective difference in the model is that Market Development Events cannot be 'switched off' and therefore the associated Effects are never temporary. Examples of possible market developments include price volatility, availability of technology, and autonomously shifting preferences.

Market Development Event	
Properties	
Property	*Specified as*
Library ID	Integer
Activation condition(s)	Logical expression
Deactivation condition(s)	Logical expression
Market Development Effects	Market Development Effect ag.set

Box B.23: Market Development Event

Market Development Effect	
Properties	
Property	*Specified as*
Library ID	Integer
Target agent(s)/object(s)	Logical expression
Value mutation	Mathematical operation

Box B.24: Market Development Effect

B.7 Links

Arriving at the last model element, Links represent relations between agents and/or objects. Service Contracts (box B.25) are the only type of Link in the model and represent seller-buyer relations. Service Contracts are dynamically created whenever a CA chooses a service-based CM and a corresponding offer on the market. They are removed when the PB no longer offers the service or when the CA reconsiders and chooses something else. Service Contracts store the price per time unit and per unit of delivered functionality at the moment when they are created. Depending on user input, the prices are either updated or not whenever the PB reconsiders its selling price.

Service Contract	
States	
State	*Specified as*
Seller	PB instance
Buyer	CB/CI instance
Price per btu	Real
Price per functional unit	Real

Box B.25: Service Contract

B.8 Simplifications and limitations

B.8.1 Excluded actors

With the distinction of PBs, CBs, CIs, World Market and Physical Environment, the model excludes two possibly relevant actors as agents in the model:

- *Government agent(s).* It may seem only natural to explicitly include Government agents that can activate policy separately from each other and have to deal with policy implementation issues. Instead, Policy Instruments can simply be switched on or off by the Observer. The reason is that the model focuses on policy *effects*, not the policy making process itself. This eliminates the need for Government agents.

- *Infrastructure Providers.* As infrastructures have been identified as a potential obstacle/barrier to large-scale adoption of servicising, it again would seem natural to include Infrastructure Providers that independently decide whether or not they want to extend their networks, thereby influencing market dynamics. They could also change connection fees. Although such agents would indeed enrich the model, it would add another layer of complexity and decision-making processes. They were not considered to be important enough by SPREE participants to justify this additional complexity. Also, little theoretical foundation was available. We have therefore decided to represent infrastructures in a similar way as skills, but with the distinction between availability and actual infrastructure connections. External market developments or policy instruments may affect the availability and prices of infrastructures in the model, otherwise these aspects are constant.

B.8.2 Business and Consumption Models

As already mentioned in section 6.6.4, the business models in the SPREE generic model are strong abstractions of reality. In the present chapter, some additional assumptions and limitations were made that affect business models:

- PBs can *only select one MM,* whereas in reality they could have multiple parallel production processes.
- PBs can *only select one product-based SM and on service-based SM* to bring the MM outputs to the market. Again, in reality they may offer, for instance, a variety of different services all based on the same product or competence.
- CAs can *only select one CM,* whereas in reality they may depend on a combination of ways to satisfy a certain Need. This is, for example, the case in the mobility case, where consumers may travel by car one day and take the train the next day. The model can still partially deal with such composite consumption models, but this requires a creative parametrisation by the user. For instance, a Consumption Model could combine characteristics of car usage and public transport usage, while still requiring car ownership as main input.
- *SM production capacity.* Only MMs have a specific production capacity: no volume restrictions are assumed for the conversion into sellable Products or Services by SMs, and the associated fixed and marginal costs are assumed volume-independent.
- Agents do not create *stocks of secondary outputs.* When these are needed in the production process, the PB always obtains them directly from the World Market in the desired quantities. There are *no volume discounts* possible.
- There are no *economies of scale.* The variable costs of an MM are independent of the chosen production capacity.
- The *price per unit of capacity of an MM is invariant.* This assumption implies that producing twice as much requires a machine twice as big, costing twice as much.
- *Structural costs* (per time unit) are constant, although real-world production facilities tend to require increasing structural (maintenance) costs as time progresses.

B.8.3 Products and Services

During the concept formalisation phase that this chapter presents the results of, we also introduced additional limitations and simplifications with respect to the model representation of products and services:

- Products do not have a *physical lifetime or expiration time,* although Tools do have a functional lifetime. Products can theoretically be infinitely stored until they are consumed.
- The only form of *contracts* included in the model are Service Contracts, and they are merely links between a seller and buyer. There are no durable product-related contracts, which are actually abundantly present in real-world supply chains.
- The only aspect of Service prices that can be changed is the price per functional unit. This excludes contract formulations where consumers pay a supplier a fixed amount of money per time unit to just 'get it done'. Such contracts can shift risk from consumers to suppliers: if 'getting it done' turns out to be very easy, the supplier makes additional profits, but if there are complications, the supplier has to pay for the additional costs.
- Specific Products and Services of the same type always have the *exact same properties* in the model, independent of the PB that produced them. This is quite a rough simplification of reality, where products from different brands are usually quite distinctive in the eyes of a consumer even when they have similar specifications.

B.8.4 Other limitations

Some final limitations introduced in the concept formalisation step include:

- CAs have a heterogeneously assigned, but still *fixed amount of Need*, every day, every season, every year. This is not true for any consuming agent in reality, and is therefore a limiting assumption. The reason for this choice is simple: agents need a predictable Need in order to be able to make calculations during decision-making processes.
- There are no skill *levels.* Skills are represented as an 'all or nothing'concept: agents either have a skill or they do not, but they cannot gradually build up a skill level or something similar.

B.9 Conclusion

This chapter has presented both visually and textually the results of the concept formalisation phase. It has identified the full list of model elements as they follow from the previous chapters. The chapter described in detail the Observer, agents, objects and links, and provided diagrams that organise these elements into a comprehensive ontology. It has discussed a number of additional limitations made here, especially with regard to business models and to the representation of products and services. The detailed description of model procedures in the next chapter will often refer back to (and make little sense without) the descriptions provided in this chapter.

C | Detailed procedures

C.1 Introduction

The previous chapter has elaborately discussed the 'what' of the SPREE generic model by presenting the ontology that has been developed for the model. Although chapter 6 has already briefly discussed the main aspects of agent behaviour and relations between them, the model procedures have not yet been presented in a formal way. This chapter presents the 'how' of the model in more detail, using pseudocode (i.e., modelling code in human-readable form) to discuss the model procedures in a more formal way. Pseudocode relates to the model formalisation step that comprises step 4 of the methodology. It is the last required step before model implementation. In this chapter, the model procedures are presented in a text-based format that offers context and some relevant calculations in the model. Appendix D contains a flowchart that represents all the subprocedures, as implemented in the model, in more detail.

Section C.2 discusses the high-level structure of model procedures that take place within one time step. Sections C.3, C.4 and C.5 elaborate on the subprocedures mentioned in section C.2. Section C.6 discusses additional limitations relating to the model formalisation step. The entire chapter depends heavily on the use of abbreviations of model elements shown in box C.1.

Abbreviations			
O:	Observer	MM:	Manufacturing Model
PB:	Producing Business	SM:	Sales Model
CB:	Consuming Business } CA	CM:	Consumption Model
CI:	Consuming Individual } CA	SK:	Skill
WM:	World Market	PP:	Policy Package
PE:	Physical Environment	PI:	Policy Instrument
TP:	Tool (Product)	PIE:	Policy Instrument Effect
CP:	Consumable (Product)	MD:	Market Development Event
S:	Service	MDE:	Market Development Effect

Box C.1: Abbreviations of model elements

C.2 Main procedures

This first section discusses the high-level structure of model procedures. Box C.2 describes the main flow of the model during one time step. At the start of the code, the Observer possibly triggers external influences on the market in the 'progress scenario' procedure. Next, PBs act by (possibly) engaging in strategic and tactical reconsiderations of their production choices. Subsequently, they produce outputs according to the choices made. CAs follow the PBs and may also engage in strategic and tactical reconsiderations. When done, they satisfy their Need for the day and restock products if necessary and applicable. Finally, the Observer increases the time step as the final preparation for the following iteration. Following sections discuss each procedure from box C.2 in more detail.

Main structure of model (sub)procedures	
O:	Progress scenario
PBs:	Strategic reconsideration of production
	Tactical reconsideration of production
	Compare new configuration and best configuration
PBs:	Tactical reconsideration of production
PBs:	Produce output
CAs:	Strategic reconsideration of consumption
	Find best offer for given CM
	Compare new CM and best CM (different for CB and CI)
CAs:	Tactical reconsideration of consumption
	Find best offer for given CM
CAs:	Satisfy Need
CAs:	Restock products
O:	Increase time step

Box C.2: The procedures that constitute the main flow in the SPREE generic model

C.3 Observer procedure

Only one Observer procedure is interesting enough to mention in this section: the 'progress scenario' procedure (box C.3). During this procedure, the Observer updates the activation of external influences related to Policy or Market Developments. Activation of a higher level element *allows for* activation of lower level elements. Deactivation of a higher level element leads to *immediate* deactivation of lower level elements. Policy Instrument Effects can be temporary and are reversed when the Policy Effect deactivates. Policy Instruments and Market Development Events do not affect properties of Products in the model that have already been produced and are owned by agents. However, they can theoretically change any property of agents, transformation models and to-be-produced Products and Services.

Progress scenario (O)
Check (de)activation conditions of PPs, and MDEs. For active PPs and MDs, check (de)activation conditions of underlying PIs and MDEs, respectively. For active PIs, check underlying PIEs If higher level structures become inactive, all lower-level structures become inactive as well For PIEs and MDEs that become active, carry out the specified effect on the market For PIEs that become inactive and have a temporary effect, reverse the specified effect on the market

Box C.3: The 'progress-scenario' procedure, where the Observer may trigger external influences on the market

C.4 Producing Business procedures

C.4.1 Strategic reconsideration of production

The Producing Business has procedures on three levels: strategic, tactical and operational. The strategic reconsideration procedure (box C.4) only takes place at agent-specific intervals. It involves an examination of several configurations of an MM and two SMs.

Strategic reconsideration of production (PB)
If no strategic reconsideration is planned for this time step, STOP. Else: Select CA-sample for market research Identify all possible configurations of MM and SM(s). For each one: Calculate total switching costs Temporarily adopt the MM and SM (no costs induced) Do a *tactical reconsideration of production* for the adopted configuration. This includes calculations of expected profit and costs. Calculate maximum threshold violation Calculate strategic fit *Compare new configuration and best configuration found so far.* If new configuration is better, it becomes the new best configuration If best configuration ≠ current configuration: Sell obsolete stocks Remove and adopt MMs and SMs where needed. Pay associated costs Adjust capacity based on the results of the tactical reconsideration

Box C.4: The model procedure for strategic reconsideration of PBs

When selecting the CA sample for market research, the PB attempts to select half the sample from her own customers, and the other half from current non-customers. This rule ensures that the CA sample is selected partially from the niche that the PB already focuses on. This allowing PBs to focus on selling

Products or Services to a small but lucrative niche in the market. In the generalisation of market research results (see below), the PB differentiates between the two groups to make sure she has a 'correct' (as far as a stochastic selection of a CA sample allows) prediction of total market demand.

There is one restriction with regard to 'possible' configurations: when a PB switches to a new MM she may only choose one SM. In a later strategic reconsideration, she can extend her business model by adding a second SM (thereby engaging in partial servicising).

The PB evaluates configurations one by one, constantly keeping the best. She takes into account switching costs, which consist of the initial investment costs of the MM and SM(s), the required additional Skills to be learned and additional network connection fees. If the PB has remaining stocks, the costs of dumping them in the PE are also included. If the stocks can be sold to the WM, the revenues count as negative switching costs.

For each configuration under consideration, the PB performs a separate tactical reconsideration procedure to identify expected maximum profits (details below). The PB also evaluates corresponding threshold violations and calculates the strategic fit of each configuration. These aspects are inputs for the subprocedure that compares two configurations (box C.5). The strategic fit is the normalised sum of (weights * scores). This is somewhat problematic if the PB considers two SMs, each with their own scores. In that case, the strategic fit is calculated as the normalised weighted average of the two individual fits, based on expected sales of both.

Compare new configuration and best configuration (PB)

Required inputs for this procedure:
  WTPf: relative willingness to pay for one additional point of strategic fit
  For both configurations a and b:
    Total expected net profit Pa and Pb
    Maximum threshold violation Va and Vb
    Strategic fit Fa and Fb: sum of (weights x scores) of each preference
1. Minimum profit check:
    If Pa $<$ 0 or Pb $<$ 0:
      Choose the option with the highest profit as best. STOP here
2. Threshold violation check:
    If Va $>$ 0 or Vb $>$ 0:
      Choose the option with the lowest violation as best. STOP here
3. Trade-off between profit and strategic fit:
    Pmax = MAX(Pa,Pb)
    If (Pb – Pa) + Pmax * WTPf * (Fb – Fa) $>$ 0:
      option b $>$ option a
    Else:
      option a $>$ option b

Box C.5: Strategic trade-off of PBs

The comparison procedure is a three-stage activity. If any configuration is unable to make a profit, the PB chooses the configuration with the highest profit (or lowest loss). If both options make a profit but either of them violates a preference threshold, the PB selects the configuration with the lowest relative threshold violation as the best option. If both options survive the threshold check, a computational comparison determines which one is best. Generally, the configuration with the highest profit is best. However, the PB is willing to sacrifice some of the (short-term) profit for a better strategic fit, thereby adhering to her long-term strategy. The difference in strategic fit score effectively counts as a bonus to the calculated profit of the corresponding configuration option. The bonus is calculated based on the 'willingness to pay for strategic fit' (WTPf) state of the PB. The WTPf is given as the fraction of profit that the PB is willing to sacrifice for an additional 'point' for the strategic fit.

An example will illustrate how this works in practice. Assume two business model configurations A and B. The PB calculates expected profits (all in euros) of 200,000 and 250,000 per year, respectively. The calculated strategic fits (after summing weights and scores on each preference aspect and then normalising back to a scale from 1 to 10) are 7.5 (A) and 6 (B). If the PB has a wtp of 0.2 (20%), it assigns to configuration A a virtual 'bonus profit' of 250,000 (i.e., maximum possible profit) * (7.5 - 6) * 0.2 = 75,000. Configuration A now has a virtual profit of 275,000 and is thus considered more attractive than configuration B.

After evaluation of all relevant configurations, the PB knows what the best option is. If the best configuration is different from the current configuration, the PB has to actually switch and pay for all switching costs identified above. The strategic reconsideration of production is always concluded by setting the production capacity to a level somewhat above the expected sales (identified during tactical reconsideration), taking into account the 'capacity over expected sales ratio' state of the PB.

C.4.2 Tactical reconsideration of production

The tactical reconsideration of production involves the optimisation of profit by setting the best output price(s) for the offer(s) produced by the current business model configuration of the PB. As box C.6 schematically illustrates, the PB constructs a demand curve based on market research. For each point on the demand curve, the PB calculates the total associated revenues (price * volume) and costs (including switching costs, costs of all inputs and outputs, variable costs). The point on the demand curve that yields the highest profits defines the price that the PB will sell her Product or Service for on the market.

If the PB has two SMs, sales gained at one SM may well partially decrease sales of the other SM, especially if the offered Product and Service have similar properties. When reconsiderating the price of one SM, the PB therefore takes into account the lost sales at the other SM. She iteratively changes price levels of both

Tactical reconsideration of production (PB)

If no tactical reconsideration is planned for this time step, STOP here
Select CA-sample for market research (or use the sample identified by the strategic reconsideration procedure)
For each SM (repeat until convergence if the PB has two SMs):
  Calculate total variable costs per unit
  Carry out market research
    Ask all CAs in CA-sample what they currently consider as the best offer on the market, and at what price they would switch from that offer to the offer the PB now considers to bring to the market
    Construct a demand curve from the CA responses, extrapolating the results to the whole population of CAs
  For each price point on the demand curve, calculate:
    Total costs made over the return on investment period, given production of the demanded volume at that price point
    Lost sales for the other SM of this PB (if any)
    Expected sales for this SM
    Expected profit
Set the SM offer prices such that they maximise profit
Update target stock levels of SMs
Update relative allocation of MM outputs to SMs

Box C.6: Tactical reconsideration of PBs

SMs until the aggregate profit no longer increases.

When the PB has identified the prices that maximise her profit, she sets the prices accordingly. The PB also updates target stock levels. The target stock level calculates as the squared root of expected sales, multiplied by a user-defined multiplier and rounded up. The square root relates to the statistical property that the variation of a stock level over time is a function of the square root of the stock size.

C.4.3 Operational rules: production of outputs

The production of outputs (box C.7) is a relatively straightforward process, consisting of buying the right amounts of inputs and getting rid of all secondary outputs. The only tricky part here is when the PB has two SMs and cannot produce enough (restricted by capacity) to replenish up to the desired stock levels of both SMs. In that case, the available production volume is allocated relative to the aggregate production shortage (over the whole time period since the last strategic reconsideration) of each SM.

Produce outputs (PB)
Calculate requested production of owned SMs as the (positive) difference between the target level and the actual level Total requested production Pr = sum of requests from individual SMs Actual production Pa is then min(Pr,Pc) (Pc = production capacity) Produce MM outputs Calculate required amounts of inputs and buy them from the WM Calculate the amounts of secondary outputs. If possible, sell them to the WM. Otherwise, dump them in the PE and pay any associated costs Shift MM outputs to SM. If Pa < Pr: Allocate based on relative total production shortage of both SMs Allocate products to main input stocks of SMs Update total production shortage of SMs Produce SM outputs (only upon consumer request if service-based) Convert inputs to the desired amount of main SM output Calculate amounts of secondary inputs and buy them from the WM Calculate amounts of secondary outputs. If possible, sell them to the WM. Otherwise, dump them in the PE and pay any associated costs If product-based: store produced main outputs in output stock

Box C.7: Production of outputs by PBs

C.5 Consuming Agent procedures

C.5.1 Strategic reconsideration of consumption

Similar to the PBs, CAs have strategic reconsiderations, tactical reconsiderations and 'operational' behaviour, which in their case constitutes satisfaction of their Need. On all three levels the procedures of CA are very comparable to the corresponding PB procedures, although Consumers have fewer options to choose from.

The strategic reconsideration of consumption (box C.8) can be triggered by any of the following conditions:

- The predefined interval between strategic reconsiderations of this CA has elapsed
- The CA has a product-based CM that does not specify automatic repurchase, and the associated owned Product has no more use time left
- The CA has a service-based CM but the supplying PB can no longer provide the Service, either because she has switched or because she has run out of stock

At the start of the procedure, the CA has to identify all CMs for which any PB offers the required main input and has sufficient stock (always required) and/or capacity (if service-based). The CA then compares all available CMs one by one using the 'Compare new CM and best CM' procedure (details below). That procedure requires the associated average costs per time unit (including switching costs and costs of all inputs

and outputs), thresholds violation and strategic or lifestyle fit of the offers to compare. The CA identifies the lowest possible costs for each considered CM by looking for the cheapest offer on the market, taking into account a loyalty correction factor (see the corresponding 'Find best input offer for given CM' below).

Strategic reconsideration of consumption (CA)

If no strategic reconsideration is planned for this time step, STOP. Else:
For each available CM:
  Check if there is any corresponding offer* on the market, with sufficient stock (if product-based) or surplus capacity (if service-based). If no suitable offer is available, STOP here
  Find best input offer for given CM. The CA now knows the buying price
  Calculate total costs over the return on investment period
  If CB:
    Calculate total profit over the return on investment period
  Calculate maximum threshold violation
  Calculate strategic (if CB) or lifestyle (if CI) fit
  Compare new CM and best CM. If the new CM is better, it becomes the new best CM
If best CM found ≠ current CM:
  Sell obsolete input products of the old CM, if any
  Remove old CM, adopt new best CM, pay all associated costs
If necessary, (re)stock

* An 'offer' is a combination of an SM output type (TP/CP/S) and its selling price

Box C.8: Strategic reconsideration of CAs

The 'Compare new CM and best CM' procedure is different for CBs and CIs. For CBs, the comparison procedure described in box C.9 is practically the same as for PBs. In principal, the CB wants the CM that maximises profit, but if other CMs provide a better fit with her strategic preferences, she makes a trade-off to decide which option is best overall. The only additional consideration here (compared to PBs) is that the CB also puts a loyalty factor into the equation. If the new offer is offered by the supplier of her current Product or Service, this counts as another bonus. The bonus is calculated based on the CB's loyalty-related willingness to pay, similar to the bonus calculated for a better strategic fit.

Revisiting the example from box C.4.1, assume that A and B now represent possible CMs that a CB has to choose from. CM A has associated profits of 200,000, CM B has associated profits of 250,000 and the strategic fit scores are 7.5 and 6, respectively. The CB has a willingness to pay for fit difference of 0.2. However, the required input for CM B can be purchased from the current supplier of this CB and the input for CM A is only offered by other PBs. Assume that the CB has a willingness to pay for loyalty (again relative to maximum profit possible) of 0.15. This adds a virtual bonus profit of 250,000 * 0.15 = 37,500 for CM B, resulting in a virtual profit of 287,500. This value is higher than the virtual profit of 275,000 calculated for CM A, so the CB will choose for CM B.

Compare new CM and best CM (CB)

Required inputs for this procedure:
  WTPf: relative willingness to pay for one additional point of strategic fit
  WTPl: relative willingness to pay to stay with current supplier
  For both CMs a and b:
    Total expected net profit Pa and Pb
    Maximum threshold violation Va and Vb
    Strategic fit Fa and Fb: sum of (weights x scores) of each preference
    Loyalty factor La and Lb (1 if supplier = current supplier, else 0)
1. Minimum profit check:
    If Pa < 0 or Pb < 0:
      Choose the CM with the highest profit as best. STOP here
2. Threshold violation check:
    If Va > 0 or Vb > 0:
      Choose the CM with the lowest violation as best. STOP here
3. Trade-off between profit and strategic fit:
    Pmax = MAX(Pa,Pb)
    If (Pb - Pa) + Pmax * (WTPf * (Fb - Fa) + WTPl * (Lb - La)) > 0:
      CM b > CM a
    Else:
      CM a > CM b

Box C.9: Strategic trade-off of CBs

For CIs, the comparison (see box C.10 has the same structure but a different goal as the procedure for CBs. That is, the CI primarily focuses on minimising costs instead of maximising profits. The CI considers the 'bonuses' relating to the lifestyle fit or loyalty as a discount on the (lowest available) costs per time unit, instead of virtual additional profit.

Completing the strategic reconsideration described in box C.8, the CA will make a switch to the newly selected CM if needed. She pays all associated switching costs including initial investment costs, skill learning costs, infrastructure connection fees and the dump costs of Products belonging to the previous CM. The dump costs count as negative switching costs if the Product can be sold to the World Market. Finally, if the selected CM is product-based, the CA buys the associated Product from the already identified best supplier.

C.5.2 Tactical reconsideration of consumption

The tactical reconsideration of consumption (box C.11) comprises of just a subset of the strategic reconsideration subprocedures, as it does not allow the CA to switch to a new CM. It's just a periodic check whether there is a similar offer (i.e., for the same CM) on the market that is much cheaper. This is especially relevant for service-based CMs, as Services typically allow for much easier switching between suppliers.

The 'find best offer' procedure identifies the best supplier for the CM currently under consideration. During a tactical reconsideration, this is always the current CM of the CA. However, this procedure is also

Compare new CM and best CM (CI)

Required inputs for this procedure:
WTPf: relative willingness to pay for one additional point of strategic fit
WTPl: relative willingness to pay to stay with current supplier
For both CMs a and b:
Total expected costs per unit of Need Ca and Cb
Maximum threshold violation Va and Vb
Lifestyle fit Fa and Fb: sum of (weights x scores) of each preference
Loyalty factor La and Lb (1 if supplier = current supplier, else 0)
1. Budget check:
If Ca > Cbudget or Cb < Cbudget:
Choose the CM with the lowest costs as best. STOP here
2. Threshold violation check:
If Va > 0 or Vb > 0:
Choose the CM with the lowest violation as best. STOP here
3. Trade-off between costs and lifestyle fit:
Cmin = MIN(Ca,Cb)
If (Cb - Ca) - Cmin * (WTPf * (Fb - Fa) + WTPl * (Lb - La)) < 0:
CM b > CM a
Else:
CM a > CM b

Box C.10: Strategic trade-off of CIs

Tactical reconsideration of consumption (CA)

If no strategic reconsideration is planned for this time step, STOP. Else:
Find best input offer for given CM
Switch to best offer
If necessary, restock products

Box C.11: Tactical reconsideration of CAs

used during strategic reconsideration, in which case the CM under consideration can also be a CM that the CA has not yet adopted.

Finding the best supplier (boxes C.12 and C.13) is not complicated. The CA first identifies the PBs that offer the Product or Service that correspond with the CM under consideration. If one of the offers is from the current supplier of the CA, that offer gets a profit bonus (if CB) or price discount (if CI) relative to the price of the cheapest option. After applying the correction for loyalty, the CA selects the offer with the highest associated profit or lowest price.

C.5.3 Satisfy Need

CMs specify how CAs can satisfy their Need each time step (box C.14). The CM, like the MM and the SM, represents a material or functional transformation process. If the main input is a Product, the CA will have

Find best offer for given CM (CB)

```
For each SM on the market that provides the needed input of the current
   CM, and has sufficient stocks (if product-based) or surplus capacity (if
   service-based):
   Obtain Pnew: profit when choosing for this offer (depending on price)
   Pmax = Max(Pnew,Pbest)
   Determine Lnew: 1 if same supplier as current supplier, 0 otherwise
   If (Pnew - Pbest) + Pmax * WTPl * (Lnew - Lbest) > 0:
      New offer > best offer so far
      Pbest = Pnew
      Lbest = Lnew
   Else:
      Best offer so far > new offer
```

Box C.12: CB procedure for finding the best offer for her CM

Find best offer for given CM (CA)

```
For each SM on the market that provides the needed input of the current
   CM, and has sufficient stocks (if product-based) or surplus capacity (if
   service-based):
   Obtain Cnew: costs per unit
   Cmin = MIN(Cnew,Cbest)
   Determine Lnew: 1 if same supplier as current supplier, 0 otherwise
   If (Cnew - Cbest) - Cmin * WTPl * (Lnew - Lbest) < 0:
      New offer > best offer so far
      Cbest = Cnew
      Lbest = Lnew
   Else:
      Best offer so far > new offer
```

Box C.13: CI procedure for finding the best offer for her CM

to own that Product and reduce its 'remaining use time' (if TP) or 'quantity' (if CP) according to the 'main input to main output ratio' of the CM. If the main input is a Service, the CM will specify how much of that Service (measured in functional units) is needed for a certain amount of Need (which may be specified in the same functional unit). Regardless of the main input, the transformation may also need secondary inputs (obtained from the World Market) and produce wastes (either sold to the World Market or dumped in the Physical Environment).

Consider an example CM that represents driving an owned car. The main input is that car. The required Need is a certain distance traveled, for instance, 40 km. The car has a total use time of 2000 hours. At an average speed of 40 km/h, the CA will have to decrease the use time of the car by 1 hour. Also, secondary inputs may be needed in the form of petrol gas, which the CA buys directly from the World Market. The transformation produces a certain amount of CO_2, which cannot be sold to the World Market but is dumped

(without costs) in the Physical Environment.

Satisfy Need (CA)
Consume the required amount of the main input: If product-based, this decreases TP use time or CP quantity If service-based, directly consume a service, that is then directly produced by the selling PB, using the *Produce outputs* procedure Calculate required amounts of inputs and buy them from the WM Calculate the amounts of secondary outputs. If possible, sell them to the WM. Otherwise, dump them in the PE and pay any associated costs If the main input is a TP and at end-of-life: If possible, sell the now useless TP to the WM. Otherwise, dump it in the PE and pay any associated costs If the CM has automatic restocking, restock to ensure sufficient main input for the next time step

Box C.14: Need satisfaction by CAs

C.6 Additional limitations

This chapter has provided much more detail about the model procedures and the steps that they consist of. By making it more clear what the model procedures consist of, the pseudocode and discussions in this chapter also implicitly introduce additional limitations that made little sense to mention in previous chapters but should surely be mentioned here. In particular, some limitations were introduced that relate to the policy activation procedure and to the strategic reconsideration procedure.

C.6.1 Policy-related

The 'progress scenario' procedure (see section C.3) introduces some limitations related to policy effects in the model:

- Agents have no choice but to *comply with policy*. They are always 'aware' of the policy and cannot make any decision on whether or not to comply. As a result, the model assumes that policy effects are always 100% effective, i.e, all model elements that meet the target group condition of the Policy Effect will be affected exactly as specified. Reality can be quite different: the effects of new policy (e.g. a subsidy being available) are usually not known to all people and organisations that the policy affects. And even all effects are indeed fully recognised, they can be ignored or bypassed. This is therefore a limitation of the model.

- In addition, agents in the model cannot anticipate policy - they can only respond once the policy activates. PBs that engage in a strategic reconsideration cannot take into account any effect of upcoming policy activation. In reality, businesses (especially large corporations) typically anticipate heavily on projected policy. They even try to actively change proposed policy by extensive lobbying. The model cannot take any of this into account.

C.6.2 Strategic reconsideration

The model has some additional simplifying assumptions and limitations that specifically relate to the strategic reconsideration procedures:

- Strategic choices of PBs are always supported by market research. This market research is free of charge in the model, which is clearly different in reality. This is partially justified by recognising that the model concept of 'market research' relates to a *broader activity* than just interviewing consumers. In reality, there are many more ways of finding out 'what the market wants', such as visiting conventions and seminars, reading professional journals and just having experience in the market. These concepts are not explicitly addressed by the model but implicitly included in the 'market research' activity.

- An important model assumption is that agents *can always make investments*, even if their cash balance is already negative. There are no explicit considerations regarding possible ways of financing an investment. The main reason is that the model should also be functional if the parametrisation of profits and losses is not exactly right, causing PBs to make structural losses and become completely paralysed. The model should be able, even without extensive tweaking of parameters such that the PBs make a nice profit, to still give insights in the 'delta', i.e., the relative effects of certain policies. Also, explicit reconsideration of financing aspects (especially when including risk, different types of loans, financial capital vs loans) adds another layer of complexity to the model with little potential to help understand servicising shifts.

- Agents do not discount costs to be made in the future. They simply calculate the expected costs based on current levels, without further modifications.

- Switching costs, in practice, are always dependent not only on the new business model configuration but also on the starting situation. The costs to switch from SM A to SM B can be different than going from SM C to SM B. In the model, *switching from any SM to SM B will always incur the same initial investment costs.* However, if the CA already adopted a Skill as a requirement for the current SM, the PB does not have to learn that skill again, making the switch cheaper (as it would be in reality).

C.6.3 Miscellaneous

The following two limitations are of a more generic nature:

- Agents never make explicit *end-of-life choices* in the model, e.g. between recycling or just dumping. If the World Market offers a price for the remainder of some discarded owned Product, the agent will always sell it. Otherwise, the agent will always just dump it into the Physical Environment. However, the model does allow for recycling and extended producer responsibility in relation to servicising. Business models that involve recycling will have a service-based SM, where the PB is assumed to keep ownership over the Product that will someday be recycled. The CA then just buys the Service to enjoy the functionality of the product without ever worrying about its end-of-life process.

- As a final limitation, CAs always have to buy their primary input from PBs in the model. The reason is that all relevant selling companies should be included in the modelled artificial market, otherwise some external offers would maintain price levels that are independent of the rest of the market, which is not realistic. The implication, however, is that the model is quite sensitive to the choices of PBs. If all PBs decide to quit producing a certain product, it will simply disappear from the market. Appropriate switching costs should prevent this from happening too often.

- Agent's *attributes do not change autonomously over time*, e.g. as a result of changes in age and family composition. State changes only occur as a result of policy, reconsideration, interaction and transformation, but are independent of time. This can be a problem if simulations span several decades: in reality, people would develop different preferences or Needs over such time periods. The model cannot take such developments into account.

C.7 Conclusion

This chapter has described the model procedures in more detail. It used a textual representation of the code to explain the sequence of agents' decisions and interactions in the model. The main model procedure first updates policy and market developments, then runs the PB procedures and finally the CA procedures. Both PB and CA procedures contain strategic, tactical and operational elements. Strategic reconsideration of PBs starts with profit maximisation based on market research. Strategic choices of all agents depend on a three-stage decision-making algorithm that filters on monetary (profit or budget) and non-monetary (preferences) thresholds, and then compares options using a trade-off formula that takes into account total profits (or costs), the preference 'fit' and, in the case of CAs, a loyalty factor. The tactical reconsideration procedures

comprise of optimalisation within the current strategic choices. The operational procedures comprise mainly of material transformations and Need satisfaction.

During model formalisation, some additional limiting assumptions were introduced that specifically regard the model procedures. Most importantly, the model assumes that policy is always 100% effective and cannot be anticipated by agents. Investments are never limited by insufficient funds and the costs of strategic switches are to a large extent independent of the current state of the agent. Agents do not make explicit end-of-life decisions and CAs cannot buy their main input from the World Market, which means that their options are highly dependent of PB choices.

This completes step 4 of the methodology. The next chapter discusses how the model elements, relations and procedures have been implemented into software.

D | Flowchart

D.1 Reading guide

This appendix contains the flowchart of implemented procedures in the NetLogo model. It corresponds directly to the NetLogo code in Appendix E.

- Each block represents a (sub)procedure.
- The first two figures show the full setup procedure, and the first level of the main 'go' procedure.
- All following figures work out one of the procedures at the first level of the go procedure.
- All procedures are only worked out in full at one place. If such procedures are also used somewhere else, a dashed line around the box indicates that the procedure has subprocedures that are included elsewhere.
- The color of a box indicates the type of agent or object that can 'run' the code, according to the explanation in the legend.

D.2 Setup

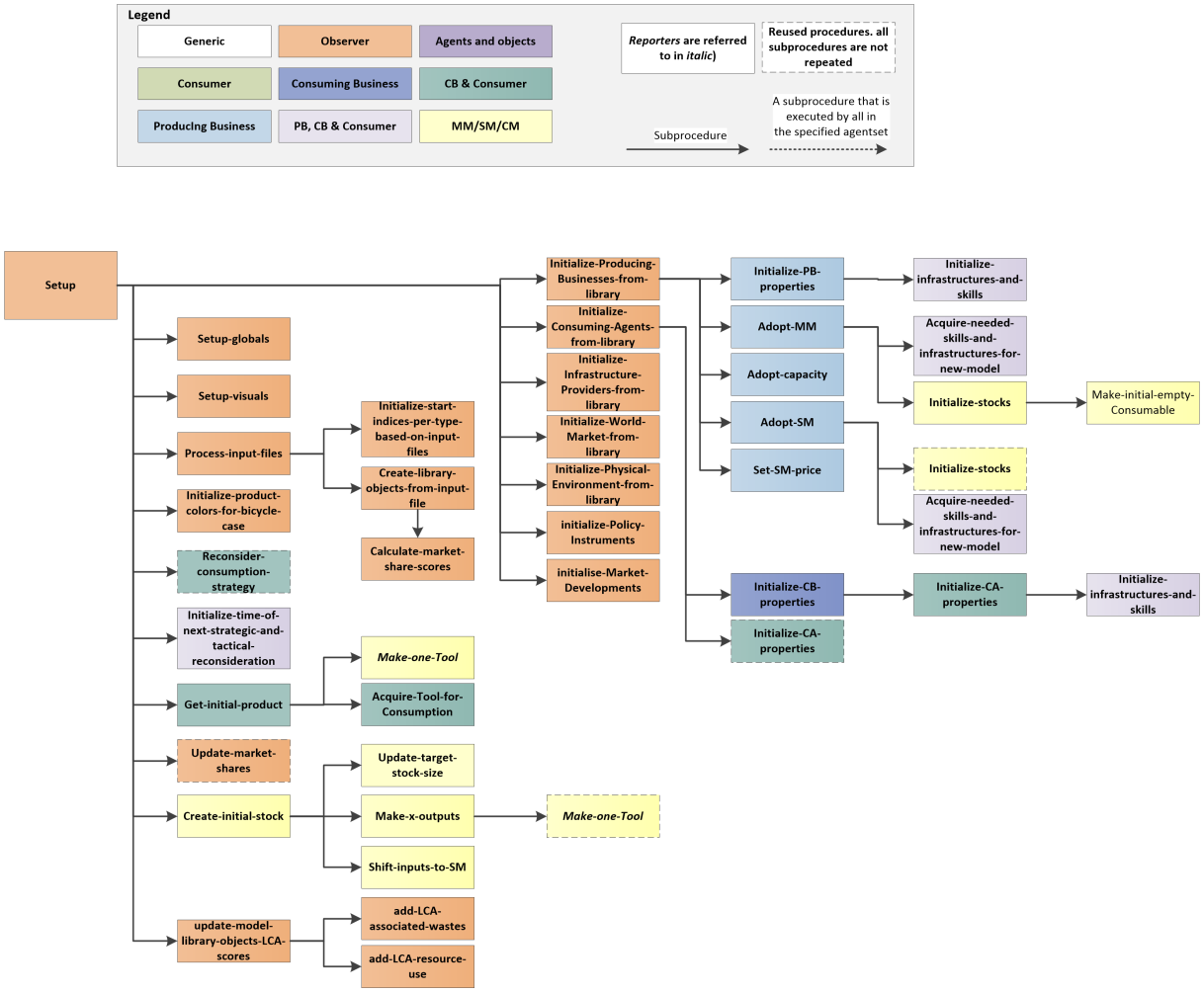

Figure D.1: Flowchart for 'setup'

D.3 Go

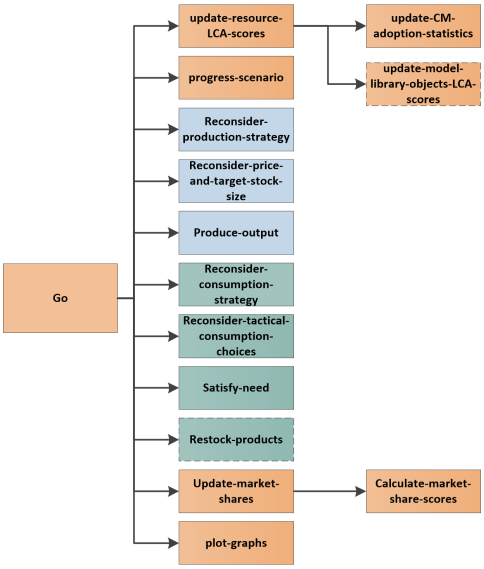

Figure D.2: Flowchart for 'go' (i.e., the main prodedure)

D.4 Reconsider-production-strategy

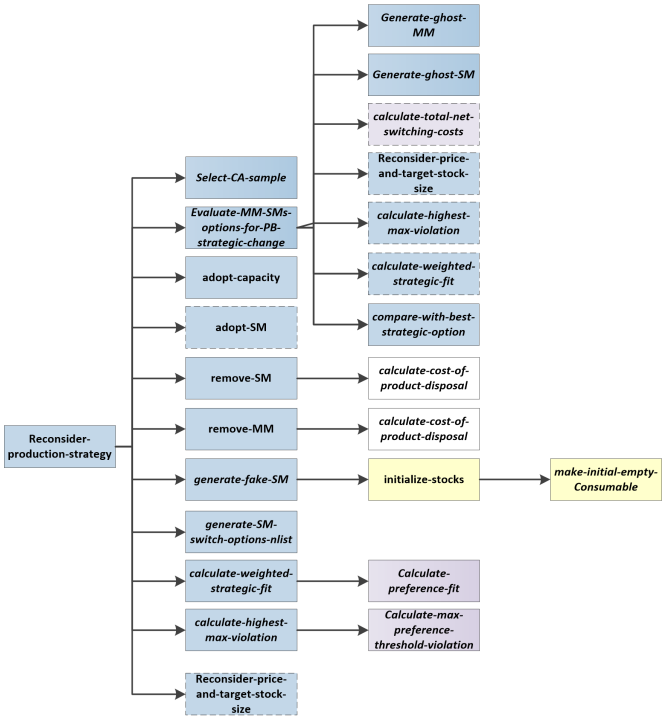

Figure D.3: Flowchart for 'reconsider-production-strategy'

D.5 Reconsider-price-and-target-stock-size

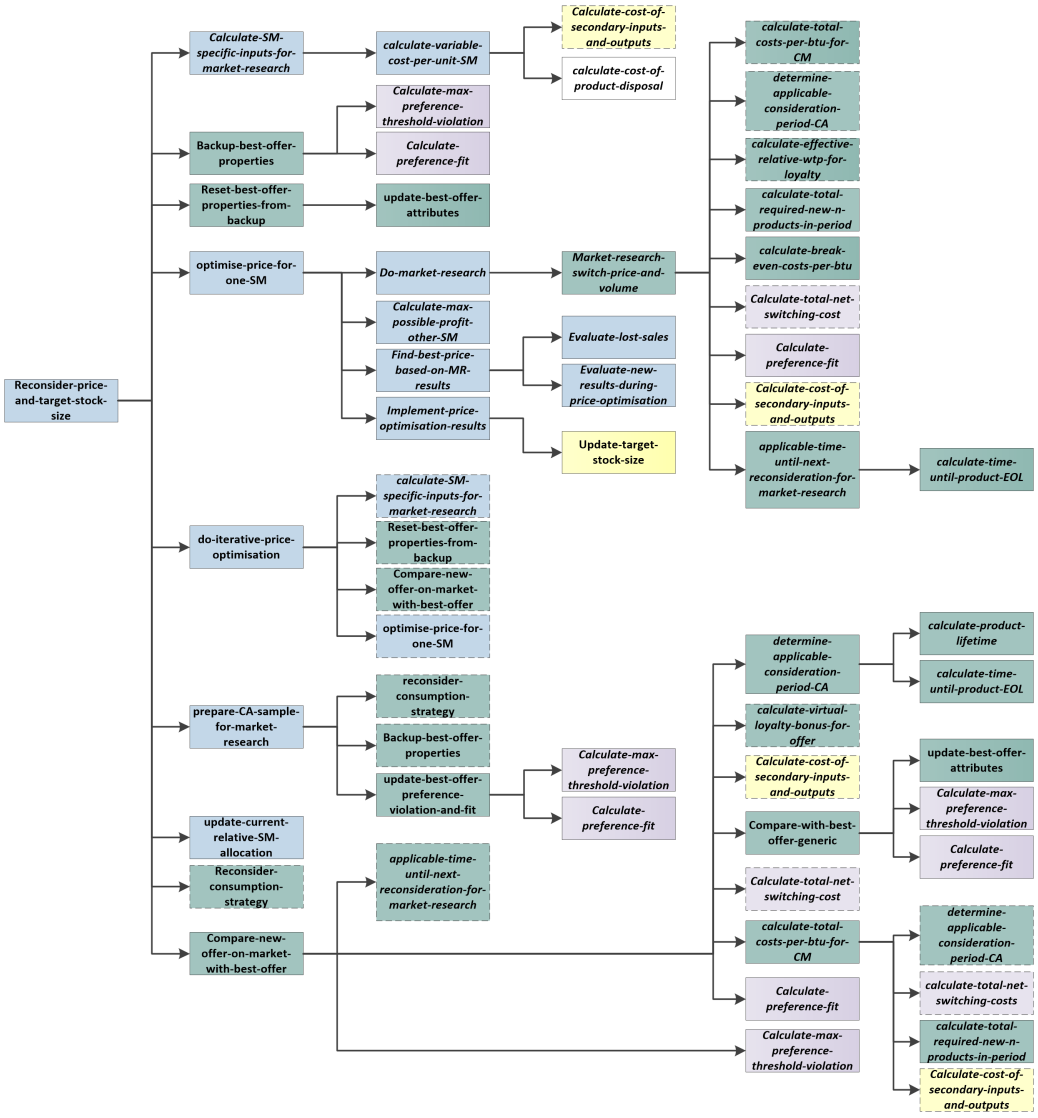

Figure D.4: Flowchart for 'reconsider-price-and-target-stock-size'

D.6 Produce-output

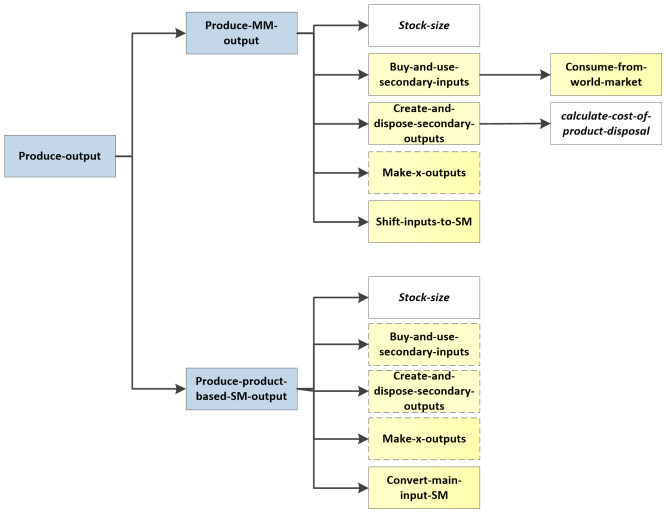

Figure D.5: Flowchart for 'produce-output'

D.7 Reconsider-consumption-strategy

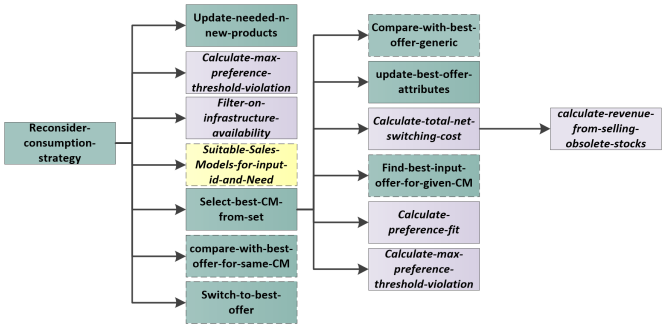

Figure D.6: Flowchart for 'reconsider-consumption-strategy'

D.8 Reconsider-tactical-consumption-choices

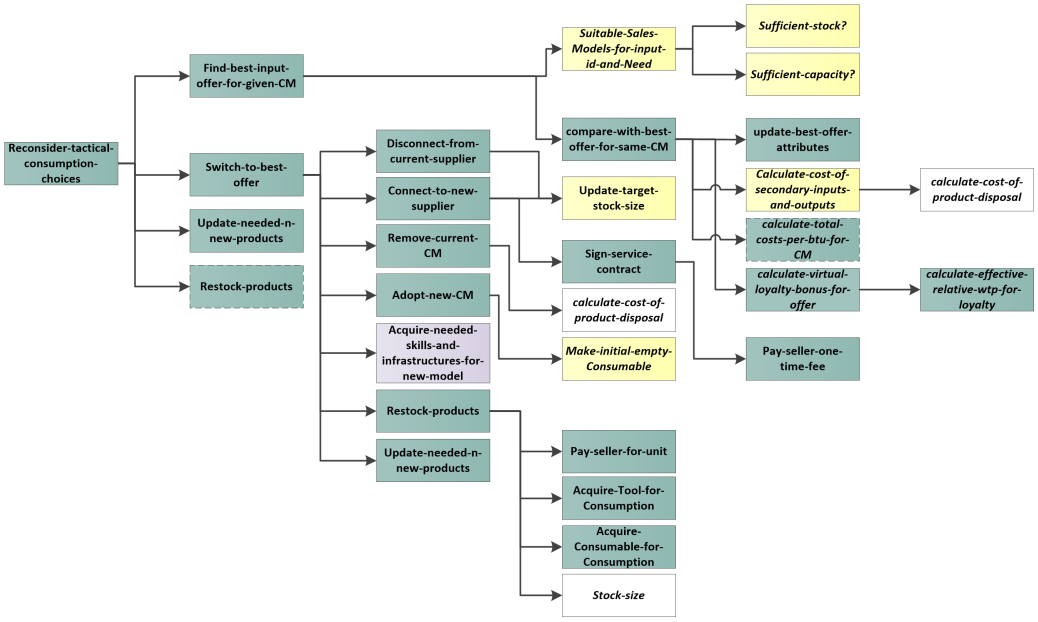

Figure D.7: Flowchart for 'reconsider-tactical-consumption-choices'

D.9 Satisfy-need

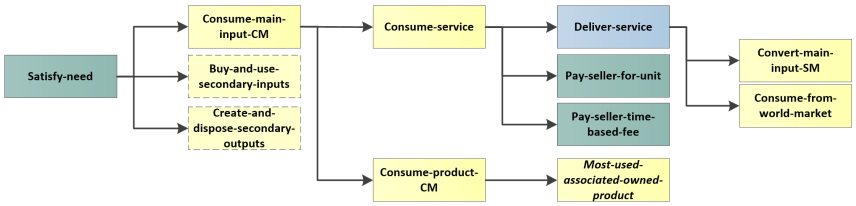

Figure D.8: Flowchart for 'satisfy-need'

E | Implemented model code

Reading guide

Procedures, arguments and reporting procedures

- The code is built up in separate procedures, each of which accomplishing a certain task and relying on other (sub)procedures to perform subtasks.
- A procedure always start with 'to' or 'to-report' (see a few lines down) and ends with 'end'
- The pseudocode flowchart (separate file) illustrates which procedures exist in the model and how they are structured.
- Procedures run from the perspective of either agents or the observer. The perspective dictates what kind of variables, arguments and subprocedures are allowed to use.
- Some procedures have 'arguments', i.e. control variables that determine how, or based on what information, they perform their task.
- Arguments are defined directly after the name of a procedure, between square brackets, and the procedure cannot run without them once defined.
- Some procedures calculate something and 'report' it back to the calling procedure. Such procedures start with 'to-report' and end with 'report <some value to report>'
- The model uses different ways to store the value of variables (i.e. globals, attributes and local variables). Examples include numerical values, strings, booleans, lists, tables, turtles and turtle-sets

NetLogo language

- NetLogo defines a number of keywords that can be used in the code (i.e. within the scope of procedures, between 'to(-report)' and 'end').
- The NetLogo dictionary (open with F1) provides an overview of those keywords and the way to use them. Pressing F1 while selecting a certain keyword brings you directly to the relevant section in the Dictionary.
- Some important keywords:
- 'ask' asks a group of agents to sequentially perform the specified task (thereby temporarily switching perspective to that of the agents in question),
- 'let' intialises a local variable (i.e. not a global or a turtle attribute)
- 'set' changes the value of a global variable, an attribute or a local variable
- 'with' allows to select only those turtles that meet the specified criteria. This keyword also temporarily changes perspective.
- 'of' allows to retrieve the value of an attribute of the specified turtle (or turtle-set, in which case the outcome is structured as a list).
- This keyword also temporarily changes perspective (to that of the specified agent)
- '(list)', 'lput', 'fput', 'replace-item' and 'sort' allow to initialise and manipulate lists, storing data in a structured way
- 'ifelse-value' checks a certain condition, then reports spefified value 1 if the condition proves true and specified value 2 if the condition proves false

Programming practices

The book by Van Dam, Nikolic and Lukszo suggests a number of good programming practices. Most of these have been incorporated during the SPREE model development. The model was (and is) stored in an SVN environment (see http://subversion.apache.org/ for the used software) that was kept up-to-date on a daily basis. Code documentation was updated at least on a weekly basis, sometimes more frequent. The model code also features a set of naming conventions:

- Uppercase/lowercase (no prefix):

  - Agents and Objects: capitalise first letter
  - Local variables: lowercase

- Prefixes:
  - 'g_' for global variables, except for two sets of globals used during specific model procedures (thereby reducing the need to communicate the values of local variables across multiple sublevels of these procedures):
    - 'opt_' for a set of global variables used exclusively for price optimisations
    - 'rec_' for a set of global variables used exclusively for PB strategic reconsiderations
  - 'a_' for attributes (i.e., states and properties):

- Suffixes:
  - '-lib' for library objects
  - '-list' to the names of any variables that store data in the form of a list
  - '-nlist' for lists within lists (i.e. nested lists)
  - '-table' for tables
  - '?' for names of variables that store data as a Boolean (i.e. true/false)

Furthermore, as instructed by Van Dam, Nikolic and Lukszo, attribute names are often quite long to make them more descriptive, even though NetLogo does not feature code completion. This helps in the tracing of bugs and in communicating the model to others.

The model elements have many more properties (states) than the ones discussed in chapter 7. All properties that have no direct match with the tables in chapter 7 are additional variables that were introduced for technical reasons, for example to store information over a longer amount of time and to facilitate decision making procedures.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;                                              ;;;;;
;;;;;                Declarations                  ;;;;;
;;;;;                                              ;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

extensions [ graphics   table   ] ;profiler

; primary agents and objects (as well as library objects to store metadata). The numbers represent turtle-type IDs (often used in the code)

; 0
breed [Resources-lib Resource-lib]
; 1
breed [Producing-Businesses-lib Producing-Business-lib]
breed [Producing-Businesses Producing-Business]
; 2
breed [Consuming-Businesses-lib Consuming-Business-lib]
breed [Consuming-Businesses Consuming-Business]
; 3
breed [Consumers-lib Consumer-lib]
breed [Consumers Consumer]
; 4
breed [World-Markets-lib World-Market-lib]
breed [World-Markets World-Market]
; 5
breed [Infrastructure-Providers-lib Infrastructure-Provider-lib]
breed [Infrastructure-Providers Infrastructure-Provider]
; 6
breed [Physical-Environments-lib Physical-Environment-lib]
breed [Physical-Environments Physical-Environment]
; 7
breed [Tools-lib Tool-lib]
breed [Tools Tool]
; 8
breed [Consumables-lib Consumable-lib]
breed [Consumables Consumable]
; 9
breed [Services-lib Service-lib]
breed [Services Service]
; 10
breed [Manufacturing-Models-lib Manufacturing-Model-lib]
breed [Manufacturing-Models Manufacturing-Model]
; 11
breed [Sales-Models-lib Sales-Model-lib]
breed [Sales-Models Sales-Model]
; 12
breed [Consumption-Models-lib Consumption-Model-lib]
breed [Consumption-Models Consumption-Model]


; Ancillary library objects to model things like skills, policy effects and market developments

; 13
breed [Skills-lib Skill-lib] ; Skills are just library objects, we don't make instances of those. We still need them as objects and not just lists so
we can properly take into account policy effects
; 14
breed [Policy-Packages-lib Policy-Package-lib]
; 15
breed [Policy-Instruments-lib Policy-Instrument-lib]
; 16
breed [Policy-Effects-lib Policy-Effect-lib]
; 17
breed [Market-Developments-lib Market-Development-lib]
; 18
breed [Market-Development-Effects-lib Market-Development-Effect-lib]


; no ID (because there's no information in the input file)

breed [Service-Contracts Service-Contract]


; eye-candy

breed [Activation-Blocks Activation-Block]
breed [SPREEs SPREE]


; global variables

globals
[ ; general measuring units
  g_basic-time-unit
  g_currency

  ; useful agentsets
  g_World-Market  ; refers to the turtle that represents the World Market
  g_Physical-Environment  ; refers to the turtle that represents the Physical Environment
  g_all-consuming-agents
  g_active-producers  ; all businesses except the latent ones
  g_active-producers-list  ; same but stored as a list
  g_all-sellable-products-and-services-lib  ; all library objects of the products and services for which a consumption model library object is
available
  g_all-library-objects-sorted-list  ; sorted on library-id

  ; supportive stuff
  g_sorted-abbreviations-list
```

```
  g_sorted-plural-type-names-list
  g_sorted-singular-type-names-list
  g_sorted-first-library-index-per-type-list
  g_number-of-resources
  g_fake-MM
  g_fake-SM-P
  g_fake-SM-S
  g_last-scanned-line

  ; supporting temporary global variables for PB strategic reconsideration
  rec_MM-ID
  rec_SM-P-ID
  rec_SM-S-ID
  rec_current-MM-ID
  rec_current-SM-P-ID
  rec_current-SM-S-ID
  rec_CA-sample

  rec_best-MM-ID
  rec_best-SM-P-ID
  rec_best-SM-S-ID
  rec_best-capacity
  rec_best-price-P
  rec_best-price-S
  rec_best-volume-P
  rec_best-volume-S
  rec_best-net-profit-P
  rec_best-net-profit-S
  rec_best-profit
  rec_best-max-violation
  rec_best-fit

  ; supporting temporary global variables for PB price optimisation
  opt_SM-P
  opt_SM-S
  opt_CA-sample
  opt_horizon
  opt_capacity-fixed?
  opt_sorted-filtered-MR-results-nlist
  opt_SM-to-update
  opt_other-SM-of-PB
  opt_associated-MM
  opt_consider-other-SM?
  opt_sorted-market-research-results-nlist
  opt_current-price
  opt_offer-price-list
  opt_price
  opt_min-price
  opt_max-price
  opt_effective-input-volume
  opt_new-total-capacity
  opt_variable-cost-per-unit
  opt_this-SM-input-to-output-ratio
  opt_free-capacity
  opt_risk-factor-this-SM
  opt_risk-factor-other-SM
  opt_other-SM-input-to-output-ratio
  opt_cumulative-input-volume-for-this-SM
  opt_cumulative-input-volume-for-other-SM
  opt_gross-profit-per-MM-output-unit-of-other-SM
  opt_net-profit-of-other-SM

  opt_best-capacity-for-this-config
  opt_best-price-P-for-this-config
  opt_best-price-S-for-this-config
  opt_best-volume-P-for-this-config
  opt_best-volume-S-for-this-config
  opt_best-net-profit-P-for-this-config
  opt_best-net-profit-S-for-this-config
  opt_best-profit-for-this-config
]

; attributes of agents and objects

Resources-lib-own  ; type 0
[
  ; basics
  a_library-id
  a_id-within-type
  a_name

  ; other
  a_measuring-unit

  a_total-amount-used-as-input-this-time-step
  a_total-amount-caused-as-waste-this-time-step
]

Producing-Businesses-lib-own   ; type 1
[
  ; basics
  a_library-id
  a_id-within-type
  a_name

  ; preferences
  a_preference-weight-dists-nlist
  a_preference-min-threshold-dists-nlist
  a_relative-wtp-for-preference-fit
```

```
  ; business model
  a_MM-capacity-and-SMs-and-prices-dist
  a_capacity-over-expected-sales-ratio
  a_market-research-relative-fraction
  a_market-research-risk-factor

  ;; reconsideration
  a_ROI-period
  a_strategic-reconsideration-interval
  a_tactical-reconsideration-interval

  ; infrastructures and skills
  a_fraction-availability-of-Infrastructures-nlist
  a_fraction-connected-to-Infrastructures-nlist
  a_fraction-with-certain-skills-nlist
]

Producing-Businesses-own   ; type 1
[
  ; basics
  a_library-id
  a_id-within-type
  a_library-object
  a_name
  a_active?
  a_cash-balance
  a_SM-colors

  ; preferences
  a_preference-weights-list
  a_preference-min-thresholds-list
  a_relative-wtp-for-preference-fit

  ; business model
  a_MM
  a_product-based-SM
  a_service-based-SM
  a_capacity-over-expected-sales-ratio
  a_market-research-relative-fraction
  a_market-research-risk-factor

  ;; reconsideration
  a_ROI-period
  a_strategic-reconsideration-interval
  a_tactical-reconsideration-interval
  a_time-of-next-strategic-reconsideration
  a_time-of-next-tactical-reconsideration

  ;; infrastructures and skills
  a_available-infrastructure-ids-list
  a_connected-infrastructure-ids-list
  a_acquired-skill-ids-list

  ;; help variables
  a_fraction-of-current-customers-chosen-for-CA-sample
  a_fraction-of-current-non-customers-chosen-for-CA-sample
  a_plot-mutation  ; used to make sure that all PBs are visible in the "PB choices" graph
]

Consuming-Businesses-lib-own   ; type 2
[
  ; basics
  a_library-id
  a_id-within-type
  a_name
  a_default-revenue-per-btu-per-unit-of-need
  a_default-cost-per-btu-per-unit-of-need
  a_initial-number-of-agents

  ; need
  a_need-per-btu-dist

  ; preferences
  a_preference-weight-dists-nlist
  a_preference-min-threshold-dists-nlist
  a_relative-wtp-for-loyalty
  a_relative-wtp-for-preference-fit

  ; reconsideration
  a_ROI-period
  a_strategic-reconsideration-interval
  a_tactical-reconsideration-interval

  ; infrastructures and skills
  a_fraction-availability-of-Infrastructures-nlist
  a_fraction-connected-to-Infrastructures-nlist
  a_fraction-with-certain-skills-nlist
]

Consuming-Businesses-own   ; type 2
[
  ; basics
  a_library-id
  a_id-within-type
  a_library-object
  a_name
  a_cash-balance
  a_time-spent
```

```
  a_total-time-spent
  a_default-revenue-per-btu
  a_default-cost-per-btu
  a_actual-revenue-per-btu

  ; need
  a_need-per-btu

  ; preferences
  a_preference-weights-list
  a_preference-min-thresholds-list
  a_relative-wtp-for-loyalty
  a_relative-wtp-for-preference-fit

  ; reconsideration
  a_ROI-period
  a_strategic-reconsideration-interval
  a_tactical-reconsideration-interval
  a_time-of-next-strategic-reconsideration
  a_time-of-next-tactical-reconsideration

  ; consumption model
  a_consumption-model
  a_service-contract
  a_current-supplier
  a_current-suppliers-SM  ;; in case of automatic repurchase
  a_needed-n-new-products

  ; infrastructures and skills
  a_available-infrastructure-ids-list
  a_connected-infrastructure-ids-list
  a_acquired-skill-ids-list

  ; supportive variables to keep track of (intermediate) choices, e.g. to compare new offers on the market with
  a_best-offer-SM
  a_best-offer-costs-per-btu
  a_best-offer-virtual-loyalty-bonus
  a_best-offer-profit-per-btu
  a_best-offer-fit
  a_best-offer-preference-violation

  a_backup-best-offer-SM
  a_backup-best-offer-costs-per-btu
  a_backup-best-offer-virtual-loyalty-bonus
  a_backup-best-offer-profit-per-btu
  a_backup-best-offer-fit
  a_backup-best-offer-preference-violation

  a_intermediate-best-offer-SM-during-MR
  a_last-market-research-volume

  a_current-offer-total-costs-per-btu
  a_current-offer-fit
]

Consumers-lib-own   ; type 3
[
  ; basics
  a_library-id
  a_id-within-type
  a_name
  a_initial-number-of-agents

  ; need
  a_need-per-btu-dist
  a_max-cost-threshold-for-unit-of-Need

  ; preferences
  a_relative-wtp-for-loyalty
  a_relative-wtp-for-preference-fit
  a_preference-weight-dists-nlist
  a_preference-min-threshold-dists-nlist

  ; reconsideration
  a_ROI-period
  a_strategic-reconsideration-interval
  a_tactical-reconsideration-interval

  ; infrastructures and skills
  a_fraction-availability-of-Infrastructures-nlist
  a_fraction-connected-to-Infrastructures-nlist
  a_fraction-with-certain-skills-nlist
]

Consumers-own   ; type 3
[
  ; basics
  a_library-id
  a_id-within-type
  a_library-object
  a_name
  a_cash-balance
  a_time-spent
  a_total-time-spent

  ; need
  a_need-per-btu
  a_max-cost-threshold-for-unit-of-Need
  a_environmental-impacts-of-last-Need-fulfillment  ;; resource list
```

```
  ; preferences
  a_relative-wtp-for-loyalty
  a_relative-wtp-for-preference-fit
  a_preference-weights-list
  a_preference-min-thresholds-list

  ; reconsideration
  a_ROI-period
  a_strategic-reconsideration-interval
  a_tactical-reconsideration-interval
  a_time-of-next-strategic-reconsideration
  a_time-of-next-tactical-reconsideration

  ; consumption model
  a_consumption-model
  a_service-contract
  a_current-supplier
  a_current-suppliers-SM  ;; in case of automatic repurchase
  a_needed-n-new-products

  ; infrastructures and skills
  a_available-infrastructure-ids-list
  a_connected-infrastructure-ids-list
  a_acquired-skill-ids-list

  ; help variables
  a_best-offer-SM
  a_best-offer-costs-per-btu
  a_best-offer-virtual-loyalty-bonus
  a_best-offer-fit
  a_best-offer-preference-violation

  a_backup-best-offer-SM
  a_backup-best-offer-costs-per-btu
  a_backup-best-offer-virtual-loyalty-bonus
  a_backup-best-offer-fit
  a_backup-best-offer-preference-violation

  a_intermediate-best-offer-SM-during-MR
  a_last-market-research-volume

  a_current-offer-total-costs-per-btu
  a_current-offer-fit
]

Infrastructure-Providers-lib-own  ; type 4
[
  a_library-id
  a_id-within-type  ;; = infrastructure id
  a_name
  a_operated-infrastructure-id
  a_one-time-connection-fee
]

Infrastructure-Providers-own  ; type 4
[
  a_library-id
  a_id-within-type  ;; = infrastructure id
  a_name
  a_operated-infrastructure-id
  a_one-time-connection-fee
]

World-Markets-lib-own   ; type 5
[ a_library-id
  a_name

  ; prices
  a_offered-product-prices-table
  a_wanted-product-prices-table
]

Physical-Environments-lib-own   ; type 6
[ a_library-id
  a_name
  a_dump-prices-table
]

Tools-lib-own   ; type 7
[
  ; basics
  a_library-ID
  a_id-within-type
  a_name
  a_default-values

  ; use-related
  a_total-use-time

  ; resource-related
  a_use-of-resources  ; per unit
  a_associated-wastes ; per unit

  ; scores
  a_preference-scores-list
  a_market-share  ;; 0 to 1
  a_market-share-score  ;; relative to the PT/CT/S with the highest market share, on a scale from 0 (lowest) to 5 (highest). Based on fraction of all
units of Need satisfied through that PT/CT/S
```

```
]

Tools-own
[
  ; basics
  a_library-id
  a_id-within-type
  a_name
  a_owner
  a_library-object

  ; use-related
  a_remaining-use-time

  ; resource-related
  a_use-of-resources  ; per unit
  a_associated-wastes ; per unit

  ; scores
  a_preference-scores-list
]

Consumables-lib-own   ; type 8
[
  ; basics
  a_library-ID
  a_id-within-type
  a_name
  a_default-values
  a_measuring-unit

  ; resource-related
  a_use-of-resources  ; per measuring unit
  a_associated-wastes ; per measuring unit

  ; scores
  a_preference-scores-list
  a_market-share  ;; 0 to 1
  a_market-share-score  ;; relative to the PT/CT/S with the highest market share, on a scale from 0 (lowest) to 5 (highest). Based on fraction of all
units of Need satisfied through that PT/CT/S
]

Consumables-own
[
  ; basics
  a_library-id
  a_id-within-type
  a_name
  a_owner
  a_library-object

  ; use-related
  a_quantity

  ; resource-related
  a_use-of-resources  ; per measuring unit
  a_associated-wastes ; per measuring unit

  ; scores
  a_preference-scores-list
]

Services-lib-own   ; type 9
[
  ; basics
  a_library-ID
  a_id-within-type
  a_name
  a_default-values
  a_functional-unit

  ; resource-related
  a_use-of-resources  ; per measuring unit
  a_associated-wastes ; per measuring unit

  ; scores
  a_preference-scores-list
  a_market-share  ;; 0 to 1
  a_market-share-score  ;; relative to the PT/CT/S with the highest market share, on a scale from 0 (lowest) to 5 (highest). Based on fraction of all
units of Need satisfied through that PT/CT/S
]

Manufacturing-Models-lib-own   ; type 10
[
  ; basics
  a_library-ID
  a_id-within-type
  a_name
  a_available?
  a_default-values
  a_associated-product-based-SM-id
  a_possible-service-based-SM-ids
  a_main-output-is-Tool?
  a_main-output-is-Consumable?
  a_possible-capacities
  a_structural-costs-per-unit-of-capacity ; incl investment, maintenance and other out-of-model-scope structural costs
  a_initial-investment-cost
  a_other-variable-costs-per-unit
  a_required-skill-ids-list
```

```
  a_required-infrastructure-ids-list

  ; inputs and outputs, per unit of main output
  a_main-output-ID
  a_main-output-library-object
  a_secondary-input-IDs-list  ;; for MMs all inputs are 'secondary', but this naming makes it consistent with other types of models
  a_secondary-input-ratios-list
  a_secondary-output-IDs-list
  a_secondary-output-ratios-list
  a_labour-time

  ;; LCA
  a_LCA-resources-use-per-unit-of-output-list
  a_LCA-associated-wastes-per-unit-of-output-list
]

Manufacturing-Models-own
[
  ; basics
  a_library-id
  a_id-within-type
  a_name
  a_owner
  a_library-object
  a_associated-product-based-SM-id
  a_possible-service-based-SM-ids
  a_main-output-is-Tool?
  a_main-output-is-Consumable?
  a_structural-costs-per-btu ; incl investment, maintenance and other out-of-model-scope structural costs. Is calculated as a factor over the chosen
capacity.
  a_other-variable-costs-per-unit
  a_required-skill-ids-list
  a_required-infrastructure-ids-list

  ; inputs and outputs, per unit of main output
  a_main-output-ID
  a_main-output-library-object
  a_secondary-input-IDs-list  ;; for MMs all inputs are 'secondary', but this naming makes it consistent with other types of models
  a_secondary-input-ratios-list
  a_secondary-output-IDs-list
  a_secondary-output-ratios-list

  ; instance-specific properties
  a_temporarily-disabled?
  a_main-output-stock
  a_output-capacity-per-btu
  a_overflow-output-capacity-due-to-rounding
  a_contracted-output-volume-per-btu
]

Sales-Models-lib-own   ; type 11
[
  ; basics
  a_library-ID
  a_id-within-type
  a_name
  a_available?
  a_default-values
  a_structural-costs-per-btu ; incl investment, maintenance and other out-of-model-scope structural costs
  a_associated-MM-id
  a_associated-MM-lib
  a_main-input-is-Tool?
  a_main-input-is-Consumable?
  a_main-output-is-Service?
  a_main-output-is-Tool?
  a_main-output-is-Consumable?
  a_initial-investment-cost ; only relevant for the library object

  a_other-variable-costs-per-unit
  a_required-skill-ids-list
  a_required-infrastructure-ids-list
  a_preference-scores-list

  ; inputs and outputs, per unit of main output
  a_main-input-ID  ;; this is also the type of stock that this sales model has
  a_main-input-ratio
  a_main-input-ratio-in-full-unit-equivalents
  a_main-input-library-object
  a_main-output-ID
  a_main-output-library-object
  a_secondary-input-IDs-list
  a_secondary-input-ratios-list
  a_secondary-output-IDs-list
  a_secondary-output-ratios-list
  a_labour-time

  ; in case of servicizing:
  a_recoverable-waste-IDs-list ; specified for one main input unit (only applicable if Tool)
  a_recoverable-waste-ratios-list  ; specified for one main input unit (only applicable if Tool)
  a_stock-demand-ratio
  a_use-time-multiplier-for-main-input

  ; initial prices
  a_offer-price-one-time-contract-cost
  a_offer-price-per-unit
  a_offer-price-per-btu

  ;; LCA
  a_LCA-resources-use-per-unit-of-output-list
```

```
  a_LCA-associated-wastes-per-unit-of-output-list
]

Sales-Models-own
[
  ; basics
  a_library-id
  a_id-within-type
  a_name
  a_associated-MM-id
  a_main-input-is-Tool?
  a_main-input-is-Consumable?
  a_main-output-is-Service?
  a_main-output-is-Tool?
  a_main-output-is-Consumable?
  a_structural-costs-per-btu ; incl investment, maintenance and other out-of-model-scope structural costs
  a_preference-scores-list
  a_other-variable-costs-per-unit
  a_required-skill-ids-list
  a_required-infrastructure-ids-list

  ; inputs and outputs, per unit of main output
  a_main-input-ID  ;; this is also the type of stock that this sales model has
  a_main-input-ratio
  a_main-input-ratio-in-full-unit-equivalents
  a_main-input-library-object
  a_main-output-ID
  a_main-output-library-object
  a_secondary-input-IDs-list
  a_secondary-input-ratios-list
  a_secondary-output-IDs-list
  a_secondary-output-ratios-list
  a_labour-time

  ; in case of servicizing:
  a_recoverable-waste-IDs-list ; specified for one main input unit (only applicable if Tool)
  a_recoverable-waste-ratios-list  ; specified for one main input unit (only applicable if Tool)
  a_stock-demand-ratio
  a_use-time-multiplier-for-main-input

  ; instance-specific properties
  a_owner
  a_associated-MM
  a_library-object
  a_temporarily-disabled?
  a_main-input-stock
  a_main-output-stock
  a_target-stock-size  ;; refers to the input stock size if the main output is a Service, and to the output stock size otherwise. Must be an integer
  a_catalogus-output
  a_offer-price-one-time-contract-cost
  a_offer-price-per-unit
  a_offer-price-per-btu
  a_total-production-shortage
  a_contracted-output-volume-per-btu

  ; statistics
  a_number-produced
  a_number-sold
]

Consumption-Models-lib-own     ; type 12
[
  ; basics
  a_library-ID
  a_id-within-type
  a_name
  a_available?
  a_default-values
  a_associated-SM-lib
  a_main-input-is-Tool?
  a_main-input-is-Service?
  a_main-input-is-Consumable?
  a_revenue-multiplier
  a_structural-costs-per-btu ; incl investment, maintenance and other out-of-model-scope structural costs
  a_initial-investment-cost ; only relevant for the library object
  a_other-variable-costs-per-unit
  a_required-skill-ids-list
  a_required-infrastructure-ids-list
  a_preference-scores-list

  ; inputs and outputs, per consumed unit of the Need
  a_main-input-ID
  a_main-input-ratio
  a_main-input-ratio-in-full-unit-equivalents
  a_main-input-library-object
  a_secondary-input-IDs-list
  a_secondary-input-ratios-list
  a_secondary-output-IDs-list  ;; for CMs all outputs are 'secondary', but this naming makes it consistent with other types of models
  a_secondary-output-ratios-list
  a_labour-time

  ; in case of productizing:
  a_automatic-repurchase?

  ;; LCA
  a_total-Need-satisfied-through-this-CM
  a_LCA-resources-use-per-unit-of-output-list
  a_LCA-associated-wastes-per-unit-of-output-list
]
```

```
Consumption-Models-own
[
  ; basics
  a_library-id
  a_id-within-type
  a_name
  a_owner
  a_library-object
  a_main-input-is-Tool?
  a_main-input-is-Service?
  a_main-input-is-Consumable?
  a_revenue-multiplier
  a_structural-costs-per-btu ; incl investment, maintenance and other out-of-model-scope structural costs
  a_other-variable-costs-per-unit
  a_required-skill-ids-list
  a_required-infrastructure-ids-list
  a_preference-scores-list

  ; inputs and outputs, per consumed unit of the Need
  a_main-input-ID
  a_main-input-ratio
  a_main-input-ratio-in-full-unit-equivalents    ;; use time of Tools is here translated into fraction of a Tool (relative to total use time). For
Consumables and Services, same value as a_main-input-ratio
  a_main-input-library-object
  a_secondary-input-IDs-list
  a_secondary-input-ratios-list
  a_secondary-output-IDs-list  ;; for CMs all outputs are 'secondary', but this naming makes it consistent with other types of models
  a_secondary-output-ratios-list
  a_labour-time

  ; in case of productizing:
  a_associated-owned-products
  a_automatic-repurchase?

  ; instance-specific
  a_consumed-main-input-volume-per-btu
]

Skills-lib-own  ; type 13
[
  ; basics
  a_library-id
  a_id-within-type  ;; = skill id
  a_name

  ; other
  a_learning-costs
]

Policy-Packages-lib-own  ; type 14
[
  ; basics
  a_library-id
  a_id-within-type  ;; = skill id
  a_name

  ; other
  a_activation-trigger
  a_deactivation-trigger
  a_policy-instruments-list
]

Policy-Instruments-lib-own  ; type 15
[
  ; basics
  a_library-id
  a_id-within-type
  a_name
  a_short-name

  ; other
  a_active?
  a_activation-trigger
  a_deactivation-trigger
  a_policy-effects-list
  a_activation-icon
]

Policy-Effects-lib-own  ; type 16
[
  ; basics
  a_library-id
  a_id-within-type
  a_name
  a_short-name

  ; other
  a_active?
  a_temporary-effect?
  a_target-group
  a_criterion
  a_effect
  a_activation-icon
]


Market-Developments-lib-own  ; type 17
[
```

```
  ; basics
  a_library-id
  a_id-within-type
  a_name
  a_short-name

  ; other
  a_active?
  a_activation-trigger
  a_MD-effects-list
  a_activation-icon
]

Market-Development-Effects-lib-own  ; type 18
[
  ; basics
  a_library-id
  a_id-within-type  ; 3
  a_name            ; Bicycle C innovation
  a_short-name

  ; other
  a_active?
  a_target-group    ; 7 (Tools-lib)
  a_criterion       ; [attribute, compare, value] -> [ a_type-within-ID, is equal to, 4 ]
  a_effect          ; [attribute, change, value] -> [a_available?, becomes, true]

  ; visualisation
  a_activation-icon
]

Service-Contracts-own
[
  a_seller
  a_buyer
  a_contracted-object
  a_price-per-btu
  a_price-per-unit
]

SPREEs-own
[
  hheading
  spin
]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;                                                      ;;;;;
;;;;;                    Setup                             ;;;;;
;;;;;                                                      ;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

to setup  ; observer procedure.
  ;; These things happen when 'Setup' is pressed. Initialises the model
  clear-all
  setup-globals  ;; initialises global variables
  setup-visuals  ;; sets up the model interface
  process-input-files ;; this results in library objects for everything, based on the input file
  initialise-product-colors-for-bicycle-case

  ; visualise the defined policy instruments and market developments
  initialise-Policy-Instruments
  initialise-Market-Developments

  ; create producing and consuming agents from the information in the library objects
  initialise-Producing-Businesses-from-library
  initialise-Consuming-Agents-from-library

  ;; let every consuming agent choose an initial consumption model based on offers on the market
  ask g_all-consuming-agents
  [ reconsider-consumption-strategy false 0 0   ;; remark: when consumers choose their initial CM, the production volumes of the chosen suppliers are
directly updated
    initialise-time-of-next-strategic-and-tactical-reconsideration  ;; randomise the times until consumer's reconsideration, otherwise everyone will
reconsider at the same time, leading to unexpected behaviour
    get-initial-product  ;; make sure the CB/C has a product associated with its CM (if not service-based), and randomise the remaining use time
  ]

  ask Producing-Businesses [ initialise-time-of-next-strategic-and-tactical-reconsideration ]

  update-market-shares  ;; calculates the market shares

  ;; create initial stocks for the PBs
  ask Producing-Businesses
  [ if has-product-SM? [ ask a_product-based-SM [ create-initial-stock ] ]
    if has-service-SM? [ ask a_service-based-SM [ create-initial-stock ] ]
  ]

  update-model-library-objects-LCA-scores

  reset-ticks
end

to initialise-product-colors-for-bicycle-case ; observer procedure
  ;; update the visual representation of the market shares
  let sorted-products-and-services-list sort-on [a_library-id] g_all-sellable-products-and-services-lib

  graphics:set-font "serif" "plain" 11
```

```
  let name-x 82
  let text-y 74

  ;; clear the market share area in the model dashboard
  graphics:set-fill-color [ 255 255 255 ]

  ;; calculate and display the market shares of the sellable products and services in the model
  foreach sorted-products-and-services-list
  [ ask ?
    [ if a_name = "Bicycle A"  [set color black]
      if a_name = "Bicycle B"  [set color blue]
      if a_name = "Bicycle A2"  [set color red]
      if a_name = "Steel A"  [set color orange + 2]
      if a_name = "Bicycle C"  [set color pink]
      if a_name = "Bicycle A sharing"  [set color grey]
      if a_name = "Bicycle B sharing"  [set color sky + 2]
      setxy 81 text-y
      set heading 0
      show-turtle
    ]

    let name [a_name] of ?

    ;; display the name of the product or service type
    graphics:draw-text name-x text-y "W" name  ;; "W" means align west (i.e. to the left)

    set text-y text-y - 2
  ]
end

to setup-globals  ; observer procedure
  ;; initialises global variables
  set g_stop-now? false
  set g_sorted-abbreviations-list (list)
  set g_sorted-plural-type-names-list (list)
  set g_sorted-singular-type-names-list (list)
  set g_sorted-first-library-index-per-type-list (list)
  set g_all-library-objects-sorted-list [ nobody ]
end

to setup-visuals  ; observer procedure.
  ;; Visualisation and graphs are set up
  no-display

  ;; make sure that all agents and objects appear the way we want them to appear in the model, e.g. as squares or arrows
  set-default-shape Producing-Businesses "square"  set-default-shape Consuming-Businesses "square"  set-default-shape Consumers "square"  set-default-
shape Tools "square"   set-default-shape Tools-lib "square"
  set-default-shape Consumables "circle"   set-default-shape Consumables-lib "circle"   set-default-shape Services "contract"   set-default-shape
Services-lib "contract"   set-default-shape Manufacturing-Models "arrow"
  set-default-shape Sales-Models "arrow"   set-default-shape Consumption-Models "arrow"    set-default-shape Service-Contracts "contract"   set-
default-shape Activation-Blocks "square"  set-default-shape SPREEs "spree"

  ;; initialise the drawing colors
  graphics:initialize min-pxcor max-pycor patch-size
  graphics:set-fill-color [ 255 255 255 ]
  graphics:set-stroke-color [ 0 0 0 ]
  graphics:set-text-color [0 0 0]

  ask patches [ set pcolor white ]

  draw-interface-lines
  type-interface-captions

  display
end

to draw-interface-lines
  let vline1_x 0 let vline1_y1 0  let vline1_y2 80
  graphics:draw-line (list (list vline1_x vline1_y1 )  (list vline1_x vline1_y2 ) )

  let vline2_x 100 let vline2_y1 0  let vline2_y2 80
  graphics:draw-line (list (list vline2_x vline2_y1 )  (list vline2_x vline2_y2 ) )

  let vline3_x 15 let vline3_y1 0  let vline3_y2 80
  graphics:draw-line (list (list vline3_x vline3_y1 )  (list vline3_x vline3_y2 ) )

  let vline4_x 80 let vline4_y1 0  let vline4_y2 80
  graphics:draw-line (list (list vline4_x vline4_y1 )  (list vline4_x vline4_y2 ) )

  ; horizontal lines
  let hline1_y 0  let hline1_x1 0  let hline1_x2 100
  graphics:draw-line (list (list hline1_x1 hline1_y )  (list hline1_x2 hline1_y ) )

  let hline2_y 80  let hline2_x1 0  let hline2_x2 100
  graphics:draw-line (list (list hline2_x1 hline2_y )  (list hline2_x2 hline2_y ) )

  let hline3_y 63  let hline3_x1 0  let hline3_x2 vline3_x
  graphics:draw-line (list (list hline3_x1 hline3_y )  (list hline3_x2 hline3_y ) )

  let hline4_y 34  let hline4_x1 0  let hline4_x2 vline3_x
  graphics:draw-line (list (list hline4_x1 hline4_y )  (list hline4_x2 hline4_y ) )

  let hline5_y 67  let hline5_x1 vline3_x  let hline5_x2 vline4_x
  graphics:draw-line (list (list hline5_x1 hline5_y )  (list hline5_x2 hline5_y ) )

  let hline6_y 53  let hline6_x1 vline3_x  let hline6_x2 vline4_x
  graphics:draw-line (list (list hline6_x1 hline6_y )  (list hline6_x2 hline6_y ) )

  create-SPREEs 1 [ set size 8  set heading 0 setxy ((vline3_x + vline4_x) / 2) ((hline5_y + hline6_y) / 2)  set hheading 50]
```

```
end

to type-interface-captions
  graphics:set-font "serif" "bold" 18
  let caption1_x 47  let caption1_y 78
  graphics:draw-text caption1_x caption1_y "C" "Sellers"

  let caption2_x caption1_x  let caption2_y 51
  graphics:draw-text caption2_x caption2_y "C" "Buyers"

  graphics:set-font "serif" "bold" 12

  let caption3_x 7.5  let caption3_y 61
  graphics:draw-text caption3_x caption3_y "C" "Policy Activation"

  let caption4_x caption3_x  let caption4_y 33
  graphics:draw-text caption4_x caption4_y "C" "Market Developments"

  let caption5_x 90  let caption5_y 79
  graphics:draw-text caption5_x caption5_y "C" "Products and Services"

  graphics:set-font "serif" "italic" 12

  let caption6_x 81  let caption6_y 77
  graphics:draw-text caption6_x caption6_y "W" "Name"

  let caption7_x 88  let caption7_y 77
  graphics:draw-text caption7_x caption7_y "W" "% of all consumption"
end

to process-input-files ; observer procedure
  ;; open the file called "Turtle type meta-IDs.txt". This file contains the names of all agents and objects in the model.
  file-close
  let filename "Turtle type meta-IDs.txt"
  file-open filename
  skip-lines 1

  ;; the input data are specified as a list with the following elements: [ abbreviation  plural-type-name  singular-type-name ]. Add this information
to the respective global variables in the model
  while [ not file-at-end? ]
  [ set g_last-scanned-line file-read-line
    let data read-from-string (word "[" g_last-scanned-line "]")
    set g_sorted-abbreviations-list lput item 1 data g_sorted-abbreviations-list
    set g_sorted-plural-type-names-list lput item 2 data g_sorted-plural-type-names-list
    set g_sorted-singular-type-names-list lput item 3 data g_sorted-singular-type-names-list
  ]

  file-close

  ; initialise starting indices per type by reading through the input files once
  initialise-start-indices-per-type-based-on-input-files

  ; initialise library objects
  create-library-objects-from-input-file

  file-close
end

to initialise-start-indices-per-type-based-on-input-files  ; observer procedure
  ;; go through the input file once to determine the number of different agent/object groups for each agent/object type in the model
  ;; each group will receive a unique library id. The 'g_sorted-first-library-index-per-type-list' contains the library id of the first group of each
type (minus one for technical reasons)
  ;; this needs to be done before creating the library objects, because the different agents and objects refer to each other, and otherwise we cannot
directly assign global library ids
  ;; open the big input file
  file-close
  file-open "SPREE model data input - illustrative case.txt"

  ;; initialise local variables
  let current-type-id 0   let next-first-library-index-per-type 0   let end-of-group? false   let next-type-to-read-in ""

  ;; iteratively traverse over all specified types, and for each type count the number of groups in the input file and keep track of the first group id
of each type
  while [ current-type-id < length g_sorted-singular-type-names-list ]
  [ set next-type-to-read-in replace-dashes item current-type-id g_sorted-singular-type-names-list
    set g_sorted-first-library-index-per-type-list (lput next-first-library-index-per-type g_sorted-first-library-index-per-type-list)

    if not (g_last-scanned-line = next-type-to-read-in) [ skip-file-to-line next-type-to-read-in ]
    skip-lines 1

    set end-of-group? false
    while [ not end-of-group? ]
    [ skip-file-to-line "--------------------------"    ;; the dashes signal a new group (within an agent/object type) in the input file
      skip-lines 1
      set next-first-library-index-per-type next-first-library-index-per-type + 1

      set end-of-group? not (g_last-scanned-line = "")   ;; a non-empty line after the dashes signals a new agent/object type in the input file
    ]
    set current-type-id current-type-id + 1
  ]
  file-close
end

to create-library-objects-from-input-file  ; observer procedure
  ;; this procedure reads through the entire input file and creates a library agent for each agent or object group that it encounters. It also assigns
all relevant values
  ;; open the big input file
  file-close
  file-open "SPREE model data input - illustrative case.txt"
```

```
  let current-type-id 0 ;; start with the resources (ID = 0)
  let end-of-group? false   let current-type-as-text ""   let task-add-library-objects-for-this-type ""

  ;; iteratively traverse over all specified types and groups and create the appropriate type of library agent for each group
  while [ current-type-id < length g_sorted-singular-type-names-list ]  ;; the agent/object types are specified in the input file by their singular
names, e.g. "Producing Business"
  [ set current-type-as-text replace-dashes item current-type-id g_sorted-singular-type-names-list

    ;; skip to the place in the input file where the next agent/object type definitions begin (e.g. search for "Producing Business"
    if not (g_last-scanned-line = current-type-as-text) [ skip-file-to-line current-type-as-text ]  ;; usually, the last scanned line already contains
that word, so don't skip anything else
    skip-lines 2

    ;; create a custom 'task' (a piece of NetLogo command code) based on the singular name of the agent/object type we want to read in.
    ;; The name of the task corresponds to the name of the appropriate procedure here in the NetLogo code
    set task-add-library-objects-for-this-type task [run (word "add-library-" item current-type-id g_sorted-singular-type-names-list "-from-text-
file")]

    set end-of-group? false
    while [ not end-of-group? ]
    [ run task-add-library-objects-for-this-type  ;; this part calls the appropriate procedure to read in the group definition we want to read in next

      ;; continue to the next group
      skip-file-to-line "--------------------------"
      skip-lines 1
      set end-of-group? not (g_last-scanned-line = "")  ;; if the line after the dashes is empty, this signals that there's another group for this
agent/object type
    ]

    if (current-type-id = 0) [ set g_number-of-resources count Resources-lib ]

    set current-type-id current-type-id + 1
  ]

  ;; make the library objects invisible and white
  foreach g_all-library-objects-sorted-list [ if is-agent? ? [ ask ? [ hide-turtle  set color white ] ] ]

  ;; make sure the 'g_all-sellable-products-and-services-lib' contains the library objects of all products and services with associated CMs in the
model
  let all-Products-and-Services (turtle-set Tools-lib Consumables-lib Services-lib)
  set g_all-sellable-products-and-services-lib all-Products-and-Services with [ any? Consumption-Models-lib with [ a_available? and (a_main-input-id =
[a_library-id] of myself) ] ]

  ;; calculate market share scores based on the initial market shares as specified in the input file
  calculate-market-share-scores

  file-close
end

to add-library-Resource-from-text-file   ; observer procedure. this object has turtle type 0
  skip-lines 1
  create-Resources-lib 1
  [                 set a_id-within-type read-from-string next-line-in-file
    skip-lines 2    set a_name next-line-in-file
    skip-lines 2    set a_measuring-unit next-line-in-file

    set a_library-id get-library-id 0 a_id-within-type

    add-myself-to-library-object-list
  ]
end

to add-library-Producing-Business-from-text-file   ; observer procedure. this agent has turtle type 1
  skip-lines 1
  create-Producing-Businesses-lib 1
  [                 set a_id-within-type read-from-string next-line-in-file
    skip-lines 2    set a_name next-line-in-file
    skip-lines 2    set a_MM-capacity-and-SMs-and-prices-dist read-from-string next-line-in-file
    skip-lines 2    set a_fraction-availability-of-Infrastructures-nlist read-from-string next-line-in-file
    skip-lines 2    set a_fraction-connected-to-Infrastructures-nlist read-from-string next-line-in-file
    skip-lines 2    set a_fraction-with-certain-skills-nlist read-from-string next-line-in-file
    skip-lines 2    set a_relative-wtp-for-preference-fit read-from-string next-line-in-file
    skip-lines 2    set a_preference-weight-dists-nlist read-from-string next-line-in-file
    skip-lines 2    set a_preference-min-threshold-dists-nlist read-from-string next-line-in-file
    skip-lines 2    set a_capacity-over-expected-sales-ratio read-from-string next-line-in-file
    skip-lines 2    set a_market-research-relative-fraction read-from-string next-line-in-file
    skip-lines 2    set a_market-research-risk-factor read-from-string next-line-in-file
    skip-lines 2    set a_ROI-period read-from-string next-line-in-file
    skip-lines 2    set a_strategic-reconsideration-interval read-from-string next-line-in-file
    skip-lines 2    set a_tactical-reconsideration-interval read-from-string next-line-in-file

    set a_library-id get-library-id 1 a_id-within-type
    ;; traverse the nested lists (within the big list) and convert the MM and SM group ids to global library ids. Then replace the original list by the
new list
    foreach list-indices a_MM-capacity-and-SMs-and-prices-dist  ;; list-indices just generates a list with the index of each item in a specified list,
so [0 1 2 ... n ] for a list of n+1 items
    [ ;; isolate the '?'th list from the big nested list
      let a_MM-capacity-and-SMs-and-prices item ? a_MM-capacity-and-SMs-and-prices-dist

      ;; replace individual id of MM (type id = 10) by global library id
      let MM-id-within-type item 0 a_MM-capacity-and-SMs-and-prices
      let MM-library-id get-library-id 10 MM-id-within-type
      set a_MM-capacity-and-SMs-and-prices replace-item 0 a_MM-capacity-and-SMs-and-prices MM-library-id

      ;; replace individual id of product-based SM (type-id = 11) by global library id
      let SM-id-within-type item 2 a_MM-capacity-and-SMs-and-prices
      if SM-id-within-type > -1
      [ let SM-library-id get-library-id 11 SM-id-within-type
```

```
        set a_MM-capacity-and-SMs-and-prices replace-item 2 a_MM-capacity-and-SMs-and-prices SM-library-id
      ]

      ;; replace individual id of service-based SM (type-id = 11) by global library id
      set SM-id-within-type item 4 a_MM-capacity-and-SMs-and-prices
      if SM-id-within-type > -1
      [ let SM-library-id get-library-id 11 SM-id-within-type
        set a_MM-capacity-and-SMs-and-prices replace-item 4 a_MM-capacity-and-SMs-and-prices SM-library-id
      ]

      ;; replace the old list (within the nested list) by the new, modified list with global library ids
      set a_MM-capacity-and-SMs-and-prices-dist replace-item ? a_MM-capacity-and-SMs-and-prices-dist a_MM-capacity-and-SMs-and-prices
    ]

    add-myself-to-library-object-list
  ]
end

to add-library-Consuming-Business-from-text-file   ; observer procedure. this agent has turtle type 2
  skip-lines 1
  create-Consuming-Businesses-lib 1
  [                set a_id-within-type read-from-string next-line-in-file
    skip-lines 2   set a_name next-line-in-file
    skip-lines 2   set a_initial-number-of-agents read-from-string next-line-in-file
    skip-lines 2   set a_fraction-availability-of-Infrastructures-nlist read-from-string next-line-in-file
    skip-lines 2   set a_fraction-connected-to-Infrastructures-nlist read-from-string next-line-in-file
    skip-lines 2   set a_fraction-with-certain-skills-nlist read-from-string next-line-in-file
    skip-lines 2   set a_need-per-btu-dist read-from-string next-line-in-file
    skip-lines 2   set a_relative-wtp-for-loyalty read-from-string next-line-in-file
    skip-lines 2   set a_relative-wtp-for-preference-fit read-from-string next-line-in-file
    skip-lines 2   set a_preference-weight-dists-nlist read-from-string next-line-in-file
    skip-lines 2   set a_preference-min-threshold-dists-nlist read-from-string next-line-in-file
    skip-lines 2   set a_default-revenue-per-btu-per-unit-of-need read-from-string next-line-in-file
    skip-lines 2   set a_default-cost-per-btu-per-unit-of-need read-from-string next-line-in-file
    skip-lines 2   set a_ROI-period read-from-string next-line-in-file
    skip-lines 2   set a_strategic-reconsideration-interval read-from-string next-line-in-file
    skip-lines 2   set a_tactical-reconsideration-interval read-from-string next-line-in-file

    set a_library-id get-library-id 2 a_id-within-type

    add-myself-to-library-object-list
  ]
end

to add-library-Consumer-from-text-file   ; observer procedure. this agent has turtle type 3
  skip-lines 1
  create-Consumers-lib 1
  [                set a_id-within-type read-from-string next-line-in-file
    skip-lines 2   set a_name next-line-in-file
    skip-lines 2   set a_initial-number-of-agents read-from-string next-line-in-file
    skip-lines 2   set a_fraction-availability-of-Infrastructures-nlist read-from-string next-line-in-file
    skip-lines 2   set a_fraction-connected-to-Infrastructures-nlist read-from-string next-line-in-file
    skip-lines 2   set a_fraction-with-certain-skills-nlist read-from-string next-line-in-file
    skip-lines 2   set a_need-per-btu-dist read-from-string next-line-in-file
    skip-lines 2   set a_max-cost-threshold-for-unit-of-Need read-from-string next-line-in-file
    skip-lines 2   set a_relative-wtp-for-loyalty read-from-string next-line-in-file
    skip-lines 2   set a_relative-wtp-for-preference-fit read-from-string next-line-in-file
    skip-lines 2   set a_preference-weight-dists-nlist read-from-string next-line-in-file
    skip-lines 2   set a_preference-min-threshold-dists-nlist read-from-string next-line-in-file
    skip-lines 2   set a_ROI-period read-from-string next-line-in-file
    skip-lines 2   set a_strategic-reconsideration-interval read-from-string next-line-in-file
    skip-lines 2   set a_tactical-reconsideration-interval read-from-string next-line-in-file

    set a_library-id get-library-id 3 a_id-within-type

    add-myself-to-library-object-list
  ]
end

to add-library-Infrastructure-Provider-from-text-file   ; observer procedure. this object has turtle type 4
  skip-lines 1
  create-Infrastructure-Providers-lib 1
  [                set a_id-within-type read-from-string next-line-in-file
    skip-lines 2   set a_name next-line-in-file
    skip-lines 2   set a_operated-infrastructure-id read-from-string next-line-in-file
    skip-lines 2   set a_one-time-connection-fee read-from-string next-line-in-file

    set a_library-id get-library-id 4 a_id-within-type

    add-myself-to-library-object-list
  ]
end

to add-library-World-Market-from-text-file   ; observer procedure. this object has turtle type 5
  skip-lines 1
  create-World-Markets-lib 1
  [                let offered-products-IDs-types-and-prices-list read-from-string next-line-in-file
    skip-lines 2   let wanted-products-IDs-types-and-prices-list read-from-string next-line-in-file

    set g_World-Market self
    set a_library-id get-library-id 5 1
    set a_name "World Market"

    ;; create the table with WM offer prices
    set a_offered-product-prices-table table:make
    foreach offered-products-IDs-types-and-prices-list
    [ table:put a_offered-product-prices-table (get-library-id (first ?) (item 1 ?)) (last ?) ]

    ;; create the table with WM wanted prices
```

```
    set a_wanted-product-prices-table table:make
    foreach wanted-products-IDs-types-and-prices-list
    [ table:put a_wanted-product-prices-table (get-library-id (first ?) (item 1 ?)) (last ?) ]

    add-myself-to-library-object-list
  ]
end

to add-library-Physical-Environment-from-text-file   ; observer procedure. this object has turtle type 6
  skip-lines 1
  create-Physical-Environments-lib 1
  [ let dumpable-products-IDs-types-and-prices-list read-from-string next-line-in-file

    set g_Physical-Environment self
    set a_name "Physical Environment"
    set a_library-id get-library-id 6 1

    ;; create the table with PE dump costs
    set a_dump-prices-table table:make
    foreach dumpable-products-IDs-types-and-prices-list
    [ table:put a_dump-prices-table (get-library-id (first ?) (item 1 ?)) (last ?) ]

    add-myself-to-library-object-list
  ]
end

to add-library-Tool-from-text-file   ; observer procedure. this object has turtle type 7
  skip-lines 1
  create-Tools-lib 1
  [ set a_id-within-type read-from-string next-line-in-file
    skip-lines 2   set a_name next-line-in-file
    skip-lines 5   set a_total-use-time read-from-string next-line-in-file
    skip-lines 2   let use-of-resources read-from-string next-line-in-file
    skip-lines 2   let associated-wastes read-from-string next-line-in-file
    skip-lines 2   set a_preference-scores-list read-from-string next-line-in-file
    skip-lines 2   set a_market-share read-from-string next-line-in-file

    set a_library-id get-library-id 7 a_id-within-type

;    set a_use-of-resources use-of-resources
;    set a_associated-wastes associated-wastes

    set a_use-of-resources n-values g_number-of-resources [0]
    foreach use-of-resources [ set a_use-of-resources replace-item ((first ?) - 1) a_use-of-resources (last ?) ]

    set a_associated-wastes n-values g_number-of-resources [0]
    foreach associated-wastes [ set a_associated-wastes replace-item ((first ?) - 1) a_associated-wastes (last ?) ]

    add-myself-to-library-object-list
  ]
end

to add-library-Consumable-from-text-file   ; observer procedure. this object has turtle type 8
  skip-lines 1
  create-Consumables-lib 1
  [               set a_id-within-type read-from-string next-line-in-file
    skip-lines 2   set a_name next-line-in-file
    skip-lines 5   set a_measuring-unit next-line-in-file
    skip-lines 2   let use-of-resources read-from-string next-line-in-file
    skip-lines 2   let associated-wastes read-from-string next-line-in-file
    skip-lines 2   set a_preference-scores-list read-from-string next-line-in-file
    skip-lines 2   set a_market-share read-from-string next-line-in-file

    set a_library-id get-library-id 8 a_id-within-type

;    set a_use-of-resources use-of-resources
;    set a_associated-wastes associated-wastes

    set a_use-of-resources n-values g_number-of-resources [0]
    foreach use-of-resources [ set a_use-of-resources replace-item ((first ?) - 1) a_use-of-resources (last ?) ]

    set a_associated-wastes n-values g_number-of-resources [0]
    foreach associated-wastes [ set a_associated-wastes replace-item ((first ?) - 1) a_associated-wastes (last ?) ]

    add-myself-to-library-object-list
  ]
end

to add-library-Service-from-text-file   ; observer procedure. this object has turtle type 9
  skip-lines 1
  create-Services-lib 1
  [               set a_id-within-type read-from-string next-line-in-file
    skip-lines 2   set a_name next-line-in-file
    skip-lines 5   set a_functional-unit next-line-in-file
    skip-lines 2   let use-of-resources read-from-string next-line-in-file
    skip-lines 2   let associated-wastes read-from-string next-line-in-file
    skip-lines 2   set a_preference-scores-list read-from-string next-line-in-file
    skip-lines 2   set a_market-share read-from-string next-line-in-file

    set a_library-id get-library-id 9 a_id-within-type

;    set a_use-of-resources use-of-resources
;    set a_associated-wastes associated-wastes
    set a_use-of-resources n-values g_number-of-resources [0]
    foreach use-of-resources [ set a_use-of-resources replace-item ((first ?) - 1) a_use-of-resources (last ?) ]

    set a_associated-wastes n-values g_number-of-resources [0]
    foreach associated-wastes [ set a_associated-wastes replace-item ((first ?) - 1) a_associated-wastes (last ?) ]
```

```
    add-myself-to-library-object-list
  ]
end

to add-library-Manufacturing-Model-from-text-file   ; observer procedure. this object has turtle type 10
  skip-lines 1
  create-Manufacturing-Models-lib 1
  [                set a_id-within-type read-from-string next-line-in-file
    skip-lines 2   set a_name next-line-in-file
    skip-lines 2   set a_available? read-from-string next-line-in-file
    skip-lines 2   let main-output-type-and-id read-from-string next-line-in-file
    skip-lines 2   let secondary-input-types-and-ids-and-ratios-list read-from-string next-line-in-file
    skip-lines 2   let secondary-output-types-and-ids-and-ratios-list read-from-string next-line-in-file
    skip-lines 2   set a_associated-product-based-SM-id read-from-string next-line-in-file
    skip-lines 2   set a_possible-service-based-SM-ids read-from-string next-line-in-file
    skip-lines 2   set a_structural-costs-per-unit-of-capacity read-from-string next-line-in-file
    skip-lines 2   set a_initial-investment-cost read-from-string next-line-in-file
    skip-lines 2   set a_labour-time read-from-string next-line-in-file
    skip-lines 2   set a_other-variable-costs-per-unit read-from-string next-line-in-file
    skip-lines 2   set a_required-skill-ids-list read-from-string next-line-in-file
    skip-lines 2   set a_required-infrastructure-ids-list read-from-string next-line-in-file

    ;; assign basic stuff based on the inputs
    set a_library-id get-library-id 10 a_id-within-type
    set a_available? ifelse-value (a_available? = 1) [ true] [false]
    set a_main-output-is-Tool? (first main-output-type-and-id = 7)
    set a_main-output-is-Consumable? (first main-output-type-and-id = 8)
    set a_main-output-id get-library-id (first main-output-type-and-id) (last main-output-type-and-id)
    set a_main-output-library-object library-object-with-id a_main-output-id

    ;; convert type + group ids to library ids and create a separate list with the ratios of each input/output compared to the main output
    set a_secondary-input-IDs-list map [get-library-id (first ?) (item 1 ?)] secondary-input-types-and-ids-and-ratios-list
    set a_secondary-input-ratios-list map [last ?] secondary-input-types-and-ids-and-ratios-list

    set a_secondary-output-IDs-list map [get-library-id (first ?) (item 1 ?)] secondary-output-types-and-ids-and-ratios-list
    set a_secondary-output-ratios-list map [last ?] secondary-output-types-and-ids-and-ratios-list

    ;; replace individual ids of SMs (type id = 11) by global library ids
    set a_associated-product-based-SM-id get-library-id 11 a_associated-product-based-SM-id
    set a_possible-service-based-SM-ids map [ get-library-id 11 ? ] a_possible-service-based-SM-ids

    add-myself-to-library-object-list
  ]
end

to add-library-Sales-Model-from-text-file   ; observer procedure. this object has turtle type 11
  skip-lines 1
  create-Sales-Models-lib 1
  [                set a_id-within-type read-from-string next-line-in-file
    skip-lines 2   set a_name next-line-in-file
    skip-lines 2   set a_available? read-from-string next-line-in-file
    skip-lines 2   set a_associated-MM-id read-from-string next-line-in-file
    skip-lines 2   let main-output-type-and-id read-from-string next-line-in-file
    skip-lines 2   let main-input-type-and-id-and-ratio read-from-string next-line-in-file
    skip-lines 2   let secondary-input-types-and-ids-and-ratios-list read-from-string next-line-in-file
    skip-lines 2   let secondary-output-types-and-ids-and-ratios-list read-from-string next-line-in-file
    skip-lines 2   let recoverable-Consumable-ids-and-ratios-list read-from-string next-line-in-file
    skip-lines 2   set a_stock-demand-ratio read-from-string next-line-in-file
    skip-lines 2   set a_use-time-multiplier-for-main-input read-from-string next-line-in-file
    skip-lines 2   set a_structural-costs-per-btu read-from-string next-line-in-file
    skip-lines 2   set a_initial-investment-cost read-from-string next-line-in-file
    skip-lines 2   set a_labour-time read-from-string next-line-in-file
    skip-lines 2   set a_other-variable-costs-per-unit read-from-string next-line-in-file
    skip-lines 2   set a_required-skill-ids-list read-from-string next-line-in-file
    skip-lines 2   set a_required-infrastructure-ids-list read-from-string next-line-in-file

    ;; assign basic stuff based on the inputs
    set a_library-id get-library-id 11 a_id-within-type
    set a_available? ifelse-value (a_available? = 1) [ true] [false]
    set a_associated-MM-id get-library-id 10 a_associated-MM-id
    set a_associated-MM-lib library-object-with-id a_associated-MM-id
    set a_main-input-id get-library-id (first main-input-type-and-id-and-ratio) (item 1 main-input-type-and-id-and-ratio)
    set a_main-input-ratio last main-input-type-and-id-and-ratio
    set a_main-input-library-object library-object-with-id a_main-input-id
    set a_main-input-is-Tool? (first main-input-type-and-id-and-ratio = 7)
    set a_main-input-is-Consumable? (first main-input-type-and-id-and-ratio = 8)
    set a_main-output-id get-library-id (first main-output-type-and-id) (last main-output-type-and-id)
    set a_main-output-library-object library-object-with-id a_main-output-id
    set a_preference-scores-list [a_preference-scores-list] of a_main-output-library-object  ;; copy the preferences from the library object of the
main output of the SM
    set a_main-output-is-Tool? (first main-output-type-and-id = 7)
    set a_main-output-is-Consumable? (first main-output-type-and-id = 8)
    set a_main-output-is-Service? (first main-output-type-and-id = 9)

    ;; convert type + group ids to library ids and create a separate list with the ratios of each input/output/recoverable waste
    set a_secondary-input-IDs-list map [get-library-id (first ?) (item 1 ?)] secondary-input-types-and-ids-and-ratios-list
    set a_secondary-input-ratios-list map [last ?] secondary-input-types-and-ids-and-ratios-list

    set a_secondary-output-IDs-list map [get-library-id (first ?) (item 1 ?)] secondary-output-types-and-ids-and-ratios-list
    set a_secondary-output-ratios-list map [last ?] secondary-output-types-and-ids-and-ratios-list

    set a_recoverable-waste-IDs-list map [get-library-id 7 (first ?)] recoverable-Consumable-ids-and-ratios-list   ;; Consumables have index 7
    set a_recoverable-waste-ratios-list map [last ?] recoverable-Consumable-ids-and-ratios-list

    ;; calculate main input to main output ratio
    set a_main-input-ratio-in-full-unit-equivalents a_main-input-ratio
    if a_main-output-is-Service?
    [ let full-amount-of-main-input ifelse-value (a_main-input-is-Tool?) [ ([a_total-use-time] of a_main-input-library-object) * a_use-time-multiplier-
for-main-input ] [ 1 ]
```

```
       ;; if the main input is a tool, store here the total use time of a new-made object of that Tool (is needed on the next line). If it is a
Consumable or a Service, just return '1'

      set a_main-input-ratio-in-full-unit-equivalents a_main-input-ratio / full-amount-of-main-input
        ;; Calculate here how much of the main input you need per time unit. e.g. if a Tool has 100 use time, and you need 5 of it per btu to fulfill
your need, the result here is 0.05. For Consumables or
        ;; Services, just return the quantity of the Consumable or Service that you need per btu (just divide by 1). Is needed so that in other
procedures, PBs can determine the total need of their customers.
    ]

    add-myself-to-library-object-list
  ]
end

to add-library-Consumption-Model-from-text-file   ; observer procedure. this object has turtle type 12
  skip-lines 1
  create-Consumption-Models-lib 1
  [               set a_id-within-type read-from-string next-line-in-file
    skip-lines 2   set a_name next-line-in-file
    skip-lines 2   set a_available? read-from-string next-line-in-file
    skip-lines 2   set a_automatic-repurchase? ifelse-value (read-from-string next-line-in-file = 1) [true] [false]
    skip-lines 2   let main-input-type-and-id-and-ratio read-from-string next-line-in-file
    skip-lines 2   let secondary-input-types-and-ids-and-ratios-list read-from-string next-line-in-file
    skip-lines 2   let secondary-output-types-and-ids-list read-from-string next-line-in-file
    skip-lines 2   set a_revenue-multiplier read-from-string next-line-in-file
    skip-lines 2   set a_structural-costs-per-btu read-from-string next-line-in-file
    skip-lines 2   set a_initial-investment-cost read-from-string next-line-in-file
    skip-lines 2   set a_labour-time read-from-string next-line-in-file
    skip-lines 2   set a_other-variable-costs-per-unit read-from-string next-line-in-file
    skip-lines 2   set a_required-skill-ids-list read-from-string next-line-in-file
    skip-lines 2   set a_required-infrastructure-ids-list read-from-string next-line-in-file

    ;; assign basic stuff based on the inputs
    set a_library-id get-library-id 12 a_id-within-type
    set a_available? ifelse-value (a_available? = 1) [ true] [false]
    set a_main-input-is-Tool? (first main-input-type-and-id-and-ratio = 7)
    set a_main-input-is-Consumable? (first main-input-type-and-id-and-ratio = 8)
    set a_main-input-is-Service? (first main-input-type-and-id-and-ratio = 9)
    set a_main-input-id get-library-id (first main-input-type-and-id-and-ratio) (item 1 main-input-type-and-id-and-ratio)
    set a_associated-SM-lib one-of Sales-Models-lib with [a_main-output-id = [a_main-input-id] of myself ]
    set a_main-input-library-object library-object-with-id a_main-input-id
    set a_main-input-ratio last main-input-type-and-id-and-ratio
    set a_preference-scores-list [a_preference-scores-list] of a_main-input-library-object

    ;; convert type + group ids to library ids and create a separate list with the ratios of each input
    set a_secondary-input-IDs-list map [get-library-id (first ?) (item 1 ?)] secondary-input-types-and-ids-and-ratios-list
    set a_secondary-input-ratios-list map [last ? ] secondary-input-types-and-ids-and-ratios-list

    set a_secondary-output-IDs-list map [get-library-id (first ?) (last ?)] secondary-output-types-and-ids-list
    set a_secondary-output-ratios-list map [last ?] secondary-output-types-and-ids-list

    ;; calculate main input to Need ratio
    let full-amount-of-main-input ifelse-value (a_main-input-is-Tool?) [ ([a_total-use-time] of a_main-input-library-object) ] [ 1 ] ;; if the main
input is a tool, store here the total use time
       ;; of a new-made object of that Tool (is needed on the next line). If it is a Consumable or a Service, just return '1'

    set a_main-input-ratio-in-full-unit-equivalents a_main-input-ratio / full-amount-of-main-input
      ;; Calculate here how much of the main input you need per time unit. e.g. if a Tool has 100 use time, and you need 5 of it per btu to fulfill
your need, the result here is 0.05. For Consumables or
       ;; Services, just return the quantity of the Consumable or Service that you need per btu (just divide by 1). Is needed so that in other
procedures, PBs can determine the total need of their customers.

    add-myself-to-library-object-list
  ]
end

to add-library-Skill-from-text-file   ; observer procedure. this object has turtle type 13
  skip-lines 1
  create-Skills-lib 1
  [               set a_id-within-type read-from-string next-line-in-file
    skip-lines 2   set a_name next-line-in-file
    skip-lines 2   set a_learning-costs read-from-string next-line-in-file

    set a_library-id get-library-id 13 a_id-within-type

    add-myself-to-library-object-list
  ]
end

to add-library-Policy-Package-from-text-file   ; observer procedure. this object has turtle type 14
  skip-lines 1
  create-Policy-Packages-lib 1
  [               set a_id-within-type read-from-string next-line-in-file
    skip-lines 2   set a_name next-line-in-file
    skip-lines 2   set a_activation-trigger read-from-string next-line-in-file
    skip-lines 2   set a_deactivation-trigger read-from-string next-line-in-file
    skip-lines 2   set a_policy-instruments-list read-from-string next-line-in-file

    set a_library-id get-library-id 14 a_id-within-type

    add-myself-to-library-object-list
  ]
end

to add-library-Policy-Instrument-from-text-file   ; observer procedure. this object has turtle type 15
  skip-lines 1
  create-Policy-Instruments-lib 1
  [               set a_id-within-type read-from-string next-line-in-file
    skip-lines 2   set a_name next-line-in-file
```

```
    skip-lines 2   set a_short-name next-line-in-file
    skip-lines 2   set a_activation-trigger read-from-string next-line-in-file
    skip-lines 2   set a_deactivation-trigger read-from-string next-line-in-file
    skip-lines 2   set a_policy-effects-list read-from-string next-line-in-file

    set a_library-id get-library-id 15 a_id-within-type

    add-myself-to-library-object-list
  ]
end

to add-library-Policy-Effect-from-text-file   ; observer procedure. this object has turtle type 16
  skip-lines 1
  create-Policy-Effects-lib 1
  [               set a_id-within-type read-from-string next-line-in-file
    skip-lines 2   set a_name next-line-in-file
    skip-lines 2   set a_temporary-effect? next-line-in-file
    skip-lines 2   set a_target-group read-from-string next-line-in-file
    skip-lines 2   set a_criterion next-line-in-file
    skip-lines 2   set a_effect next-line-in-file

    set a_library-id get-library-id 16 a_id-within-type

    add-myself-to-library-object-list
  ]
end

to add-library-Market-Development-from-text-file   ; observer procedure. this object has turtle type 17
  skip-lines 1
  create-Market-Developments-lib 1
  [               set a_id-within-type read-from-string next-line-in-file
    skip-lines 2   set a_name next-line-in-file
    skip-lines 2   set a_short-name next-line-in-file
    skip-lines 2   set a_activation-trigger read-from-string next-line-in-file
    skip-lines 2   set a_MD-effects-list read-from-string next-line-in-file

    set a_library-id get-library-id 17 a_id-within-type

    add-myself-to-library-object-list
  ]
end

to add-library-Market-Development-Effect-from-text-file   ; observer procedure. this object has turtle type 18
  skip-lines 1
  create-Market-Development-Effects-lib 1
  [               set a_id-within-type read-from-string next-line-in-file
    skip-lines 2   set a_name next-line-in-file
    skip-lines 2   set a_target-group next-line-in-file
    skip-lines 2   set a_criterion next-line-in-file
    skip-lines 2   set a_effect next-line-in-file

    set a_library-id get-library-id 18 a_id-within-type

    add-myself-to-library-object-list
  ]
end

to initialise-Producing-Businesses-from-library   ; observer procedure
  let next-group-xcor 19
  let label-ycor 73

  let PB-group-IDs sort [a_library-id] of Producing-Businesses-lib
  let groupinterval 63 / (length PB-group-IDs)
  let agent-size 4

  let sorted-PB-libs sort-on [ a_library-id ] Producing-Businesses-lib

  ;; the model uses the specified colors as base colors to visualise Sales Models, as well as related Consumption Models
  let SM-base-colors [blue red yellow violet green]

  let PB-id 0

  ;; iteratively add PBs based on the information in the PB-lib turtles
  foreach sorted-PB-libs
  [ ask ?
    [ let nextx next-group-xcor
      let nexty label-ycor

      draw-interface-agent-group-label groupinterval next-group-xcor label-ycor agent-size

      ;;  align the producing businesses in this group nicely over the available space in the model interface and assign initial MM, capacity, SM and
offer price according to input data
      foreach a_MM-capacity-and-SMs-and-prices-dist
      [ let MM-capacity-and-SM-and-prices butlast ?
        let number-of-agents last ?

        hatch-Producing-Businesses number-of-agents
        [ set a_SM-colors (list ((item PB-id SM-base-colors) - 1) ((item PB-id SM-base-colors) + 2))
          initialise-PB-properties MM-capacity-and-SM-and-prices myself agent-size nextx nexty

          set nextx nextx + agent-size
          if nextx >= next-group-xcor + groupinterval - 2 * agent-size + 2
          [ set nextx next-group-xcor
            set nexty nexty - agent-size
          ]

          set a_plot-mutation -0.25 + 0.1 * PB-id

          set PB-id PB-id + 1
```

```
        ]
      ]

      set next-group-xcor next-group-xcor + groupinterval
    ]
  ]

  set g_active-producers (turtle-set Producing-Businesses with [a_active?])
  set g_active-producers-list sort-on [who] g_active-producers
end

to draw-interface-agent-group-label [ groupinterval nextgroupx groupystart agent-size ] ;; PB/CB/C procedure
  ;; display group name
  graphics:set-font "serif" "bold" 12
  let group-name a_name
  if length group-name > groupinterval * 1.5
    [ set group-name (word (substring group-name 0 (round groupinterval * 1.5) ) "...")]
  graphics:draw-text nextgroupx + ((groupinterval - (agent-size * 2) + 1) / 2) (groupystart + agent-size / 2 + 1) "C" group-name
end


to set-color-PB  ;; PB procedure
  ;; if a PB only has a product-based SM, it's shown as darker grey. Only service-based shows as light grey. A combination has a color in-between.
  let color-id ifelse-value is-turtle? a_product-based-SM [ 1 ] [ 0 ] + ifelse-value is-turtle? a_service-based-SM [ 2 ] [ 0 ]
  set color item color-id (list black (grey + 2) (grey + 4) (grey + 3) )
end

to set-color-CA  ;; CB/C procedure
  ;; Consuming agents with a product-based CM show as darker grey, those with a service-based SM are colored light grey
  let color-id ifelse-value ([a_main-input-is-service?] of a_Consumption-Model) [ 0 ] [ 1 ]
  set color item color-id (list (grey + 4) (grey + 2))
end

to initialise-Consuming-Agents-from-library ; observer procedure
  let next-group-xcor 17
  let label-ycor 47

  let CA-group-IDs sort [a_library-id] of (turtle-set Consumers-lib Consuming-Businesses-lib)
  let groupinterval 63 / (length CA-group-IDs)
  let agent-size 2

  ;; iteratively add consuming businesses based on the information in the CB_lib turtles
  let sorted-CB-libs sort-on [ a_library-id ] Consuming-Businesses-lib
  foreach sorted-CB-libs
  [ ask ?
    [ let nextx next-group-xcor
      let nexty label-ycor

      draw-interface-agent-group-label groupinterval next-group-xcor label-ycor agent-size

      ;;  align the consuming businesses in this group nicely over the available space in the model interface
      hatch-Consuming-Businesses a_initial-number-of-agents
      [ initialise-CA-properties myself agent-size nextx nexty
        initialise-CB-properties

        set nextx nextx + agent-size
        if nextx >= next-group-xcor + groupinterval - 2 * agent-size + 2
        [ set nextx next-group-xcor
          set nexty nexty - agent-size
        ]
      ]
      set next-group-xcor next-group-xcor + groupinterval
    ]
  ]

  let sorted-C-libs sort-on [ a_library-id ] Consumers-lib

  ;; do the same for Consumers
  foreach sorted-C-libs
  [ ask ?
    [ let nextx next-group-xcor
      let nexty label-ycor

      draw-interface-agent-group-label groupinterval next-group-xcor label-ycor agent-size

      ;;  align the consumers in this group nicely over the available space in the model interface
      hatch-Consumers a_initial-number-of-agents
      [ initialise-CA-properties myself agent-size nextx nexty

        set nextx nextx + agent-size
        if nextx >= next-group-xcor + groupinterval - 2 * agent-size + 2
        [ set nextx next-group-xcor
          set nexty nexty - agent-size
        ]
      ]
      set next-group-xcor next-group-xcor + groupinterval
    ]
  ]

  set g_all-consuming-agents (turtle-set Consuming-Businesses Consumers)  ;; store the Consuming Businesses and Consumers together in the global
variable 'g_all-consuming-agents'
end

to initialise-PB-properties [ MM-capacity-and-SM-and-prices library-object agent-size x y ] ;; PB procedure to assign initial instance-specific
properties of newly created PBs (with all general properties inherited from the library agent)
  set a_library-object library-object
  setxy x y

  ;; find all the needed distributions, all specified as (nested) lists with minimum and maximum values of a uniformly distributed space
```

```
  let preference-weight-dists-nlist [a_preference-weight-dists-nlist] of a_library-object
  let preference-min-threshold-dists-nlist [a_preference-min-threshold-dists-nlist] of a_library-object

  ;; sample from the distributions the corresponding attributes of this agent
  set a_preference-weights-list map [ random-uniform-integer (first ?) (last ?) ] preference-weight-dists-nlist   ;; something with map...
  set a_preference-min-thresholds-list map [ random-uniform-integer (first ?) (last ?) ] preference-min-threshold-dists-nlist

  initialise-infrastructures-and-skills

  ;; adopt initial MM and capacity
  adopt-MM (first MM-capacity-and-SM-and-prices)
  adopt-capacity (item 1 MM-capacity-and-SM-and-prices)

  ;; if applicable, adopt initial product-based SM and set price
  if (item 2 MM-capacity-and-SM-and-prices) > -1
    [ adopt-SM (item 2 MM-capacity-and-SM-and-prices) (item 3 MM-capacity-and-SM-and-prices ) 0
      ;; 'set-SM-price' needs to know (as first argument) whether it should set the price of the product-based or service-based SM, even if the agent
has only one (as is the case initially)
    ]

  ;; if applicable, adopt initial service-based SM and set price
  if (item 4 MM-capacity-and-SM-and-prices) > -1
    [ adopt-SM (item 4 MM-capacity-and-SM-and-prices) (item 5 MM-capacity-and-SM-and-prices)  0
      ;; 'set-SM-price' needs to know (as first argument) whether it should set the price of the product-based or service-based SM, even if the agent
has only one (as is the case initially)
    ]

  set a_active? true

  ;; appearance
  set size agent-size
  set-color-PB
  show-turtle
end

to initialise-CB-properties  ;; CB procedure to assign initial instance-specific properties of newly created CBs (with all general properties inherited
from the library agent)
  ;; calculate the default revenue and cost per btu of the CB, based on amount of Need and revenue/cost per unit of Need specified in the input data
  set a_default-revenue-per-btu a_Need-per-btu * [a_default-revenue-per-btu-per-unit-of-Need] of a_library-object
  set a_default-cost-per-btu a_Need-per-btu * [a_default-cost-per-btu-per-unit-of-Need] of a_library-object
end

to initialise-CA-properties [ library-object agent-size x y ] ;; CB/C procedure to assign initial instance-specific properties of newly created CBs/Cs
(with all general properties inherited from the library agent)
   set a_library-object library-object

  ;; find all the needed distributions, all specified as (nested) lists with minimum and maximum values of a uniformly distributed space
  let need-per-btu-dist [a_need-per-btu-dist] of a_library-object
  let preference-weight-dists-nlist [a_preference-weight-dists-nlist] of a_library-object
  let preference-min-threshold-dists-nlist [a_preference-min-threshold-dists-nlist] of a_library-object

  ;; sample from the distributions the corresponding attributes of this agent
  set a_Need-per-btu (first need-per-btu-dist) + random-float (last need-per-btu-dist - first need-per-btu-dist)
  set a_preference-weights-list map [ random-uniform-integer (first ?) (last ?) ] preference-weight-dists-nlist   ;; something with map...
  set a_preference-min-thresholds-list map [ random-uniform-integer (first ?) (last ?) ] preference-min-threshold-dists-nlist

  ;; somewhat randomise the reconsideration intervals (factor 0.8 - 1.2), to prevent all consuming agents from perfectly aligning their reconsideration
moment, which would yield unwanted results
  set a_strategic-reconsideration-interval round (a_strategic-reconsideration-interval * (0.8 + random-float 0.4))
  set a_tactical-reconsideration-interval round (a_strategic-reconsideration-interval * (0.8 + random-float 0.4))
  if a_tactical-reconsideration-interval > a_strategic-reconsideration-interval [ set a_tactical-reconsideration-interval a_strategic-reconsideration-
interval ]

  initialise-infrastructures-and-skills

  set size agent-size
  setxy x y
  show-turtle
end

to initialise-infrastructures-and-skills  ;; PB/CB/C procedure
  ;; find all the needed probabilities, all specified as (nested) lists that specify the ID of an infrastructure or skill
  ;; and then the fraction of this group of agents that has the Infrastructure availability/connectivity or the specified Skill
  let fraction-availability-of-infrastructures-nlist [a_fraction-availability-of-infrastructures-nlist] of a_library-object
  let fraction-connected-to-infrastructures-nlist [a_fraction-connected-to-infrastructures-nlist] of a_library-object
  let fraction-with-certain-skills-nlist [a_fraction-with-certain-skills-nlist] of a_library-object

  ;; initialise the lists that store the available/connected Infrastructures and acquired Skills of the agent
  set a_available-infrastructure-ids-list (list)
  set a_connected-infrastructure-ids-list (list)
  set a_acquired-skill-ids-list (list)

   ;; evaluate for each type of infrastructure mentioned in the input file whether the new agent instance has availability to that infrastructure.
   ;; if so, add the ID of the infrastructure to the 'a_available-infrastructures-list' of the agent
   ;; the nested-list is a 'list of lists'. '?' refers to each inner list, which is a combination of an infrastructure id ('first') and the fraction of
agents that has it available ('last')
  foreach fraction-availability-of-infrastructures-nlist [ if (random-float 1 <= (last ?)) [ set a_available-infrastructure-ids-list lput (first ?)
a_available-infrastructure-ids-list ] ]

  ;; now for each available infrastructure of the agent, randomly assign whether the agent actually has a connection, bsaed on the specified
probability
  foreach fraction-connected-to-infrastructures-nlist
  [ if member? (first ?) a_available-infrastructure-ids-list
    [ if (random-float 1 <= (last ?)) [ set a_connected-infrastructure-ids-list lput (first ?) a_connected-infrastructure-ids-list ] ]
  ]

  ;; Do a similar thing or Skills. The nested-list is a 'list of lists'. '?' refers to each inner list, which is a combination of a skill id ('first')
and the fraction of agents that already has it ('last')
```

```
  foreach fraction-with-certain-skills-nlist [ if (random-float 1 <= (last ?)) [ set a_acquired-skill-ids-list lput (first ?) a_acquired-skill-ids-list
] ]
end

to initialise-Policy-Instruments ; observer procedure
  ;; update the visual representation of the market shares
  let sorted-policy-instruments-list sort-on [a_library-id] Policy-Instruments-lib

  graphics:set-font "serif" "plain" 11
  let name-x 2
  let text-y 59

  ;; iteratively add a text label and an activation icon for each of the defined PIs
  foreach sorted-policy-instruments-list
  [ let name [a_short-name] of ?

    graphics:draw-text name-x text-y "W" name  ;; "W" means align west (i.e. to the left)

    ;; the activation block indicates for human observers whether the PI is currently active
    ask ?
    [ hatch-Activation-Blocks 1
      [ set color red   setxy 1 text-y   show-turtle
        ask myself [ set a_activation-icon myself ]
      ]
    ]

    set text-y text-y - 2
  ]
end

to initialise-Market-Developments ; observer procedure
  ;; update the visual representation of the market shares
  let sorted-market-developments-list sort-on [a_library-id] Market-Developments-lib

  graphics:set-font "serif" "plain" 11
  let name-x 2
  let text-y 31

  show sorted-market-developments-list

  ifelse g_MDs?
  [ ;; iteratively add a text label and an activation icon for each of the defined PIs
    foreach sorted-market-developments-list
    [ let name [a_short-name] of ?

      graphics:draw-text name-x text-y "W" name  ;; "W" means align west (i.e. to the left)

      ;; the activation block indicates for human observers whether the PI is currently active
      ask ?
      [ hatch-Activation-Blocks 1
        [ set color red   setxy 1 text-y   show-turtle
          ask myself [ set a_activation-icon myself ]
        ]
      ]

      set text-y text-y - 2
    ]
  ]
  [ graphics:set-font "serif" "italic" 11
    graphics:draw-text 1 text-y "W" "Market Developments"
    graphics:draw-text 1 text-y - 2 "W" "are not yet available!"
    graphics:set-font "serif" "plain" 11
  ]
end

to initialise-time-of-next-strategic-and-tactical-reconsideration  ;; PB/CB/C procedure
  ;; Here we initialise the remaining times until first strategic and tactical reconsideration.
  ;; These are randomised based on the respective reconsideration intervals sampled for the agents earlier during the setup
  set a_time-of-next-strategic-reconsideration random-uniform-integer 1 a_strategic-reconsideration-interval
  set a_time-of-next-tactical-reconsideration random-uniform-integer 1 a_tactical-reconsideration-interval
end

to get-initial-product  ;; CB/C procedure
  ask a_Consumption-Model
  [ ;; if the main input is a Tool, make sure the Consuming Agent at least has enough of them to satisfy her Need during the first tick. Randomise the
exact remaining use time
    if a_main-input-is-Tool?
    [ let owner a_owner
      let main-input-id a_main-input-id
      let n-th-product 1
      let min-needed-n-products a_consumed-main-input-volume-per-btu

      let min-fraction-remaining-of-first-product (min-needed-n-products mod 1) + (ifelse-value ((min-needed-n-products mod 1) = 0) [1] [0])
        ;; explanation of the formula:
        ;; mod (modulus): divide something by something else as often as you can, then return the rest value. e.g. 4.4 mod 2 --> 0.4
        ;; this whole thing is meant to ensure that the agent can satisfy its Need during the first tick.
        ;; since it always uses only the oldest Tool first, if you have multiple owned associated products for a CM..
        ;; .. only one (the oldest) is partially used, the others have all the use time remaining. For that oldest product, we here
        ;; determine how much use time the oldest product must have at a minimum to ensure that the agent can satisfy its Need in the
        ;; first tick. The actual use time then randomly lies between that minimum value, and the maximum use time that the Tool can have.
        ;; the 'ifelse-value' part is for the rare case that an exact integer number of products is needed each btu...

      ;; iteratively add the required initial number of products owned by the agent. Randomise the remaining use time as described above.
      repeat ceiling min-needed-n-products
      [ ask [a_current-suppliers-SM] of a_owner
        [ let new-product make-one-Tool  ;; make-one-output-unit is a reporting procedure that produces one unit of its main output and then returns
the produced product
```

```
          ;; if this is the first created product for this CB/C, randomise the remaining use time. Otherwise (if its the 2nd or 3rd product), the
remaining use time is automatically set to the total use-time of new Tools of that type
          ask new-product
          [ if n-th-product = 1
            [ set a_remaining-use-time a_remaining-use-time * (min-fraction-remaining-of-first-product + random-float (1 - min-fraction-remaining-of-
first-product))
              set a_remaining-use-time (ceiling (a_remaining-use-time * 10)) / 10  ;; to remove some decimals
            ]

            ask owner [ acquire-Tool-for-Consumption myself ]
          ]
        ]
        set n-th-product n-th-product + 1
      ]
    ]
    ;; if the main input is a Consumable, just make sure the consumer has exactly enough to satisfy her Need during the first tick
    if a_main-input-is-Consumable?
    [ let needed-quantity a_consumed-main-input-volume-per-btu
      ask a_associated-owned-products [ set a_quantity needed-quantity ]
    ]
  ]
  set a_needed-n-new-products 0
end

to acquire-Tool-for-Consumption [ product ]  ;; CB/C procedure. 'product' refers to a product in the stock of the SM of the supplier of the CB/C.
  ;; A product is always first created by the supplier (even during initialisation). This procedure transfers ownership to the CB/C.

  ;; remove the Tool from the supplier's SM output stock
  ask a_current-suppliers-SM [ set a_main-output-stock remove-from-turtle-set product a_main-output-stock ]

  ;; couple the product to the respective CM and CB/C
  ask a_Consumption-Model [ set a_associated-owned-products add-to-turtle-set product a_associated-owned-products ]
  ask product
  [ set a_owner myself
    visualise-product ([a_Consumption-Model] of myself) true
  ]
end

to acquire-Consumable-for-Consumption [ amount ]  ;; CB/C procedure
  ;; A product is always first created by the supplier (even during initialisation). This procedure transfers ownership to the CB/C.

  ;; reduce the supplier's SM output stock by the amount specified
  ask a_current-suppliers-SM [ ask a_main-output-stock [ set a_quantity a_quantity - amount ] ]

  ;; update the input stock of the CM
  ask a_Consumption-Model [ ask a_associated-owned-products [ set a_quantity a_quantity + amount ] ]
end

to create-initial-stock  ;; SM procedure
  ;; This procedure initialises the target stock size and initial stock of the SM, based on the initial demand
  let current-customers g_all-consuming-agents with [ a_current-suppliers-SM = myself ]

  ;; determine the long-term average amount consumed by current customers.
  let total-sales-per-btu sum [ [a_consumed-main-input-volume-per-btu] of a_Consumption-Model ] of current-customers

  update-target-stock-size total-sales-per-btu

  let initial-stock-size a_target-stock-size

  ifelse a_main-output-is-Service?
  [ ;; if the SM is service-based, create an initial INPUT stock of products, to make sure initial demand can be satisfied
    ask a_associated-MM
    [ make-x-outputs initial-stock-size
      shift-inputs-to-SM myself initial-stock-size
    ]
    set a_number-produced a_number-produced + initial-stock-size
  ]
  [ ;; if the SM is product-based, create an initial OUTPUT stock of products, to make sure initial demand can be satisfied
    make-x-outputs initial-stock-size
    set a_number-produced a_number-produced + initial-stock-size
  ]
end

to update-target-stock-size [ expected-outflow-per-btu ] ;; SM procedure
  ;; the target stock is an input stock if the SM is service-based, and an output stock if the SM is product based.
  ;; 'expected-outflow-per-btu' refers to the expected outflow of the respective stock (so either input or output).
  ;; this procedure sets the target stock size based on a formula based on 'expected-outflow-per-btu'


  if a_main-output-is-Service?
  [ set expected-outflow-per-btu expected-outflow-per-btu * a_main-input-ratio-in-full-unit-equivalents ]

  ;; the sqrt is because at larger numbers, the correction needed will be lower because the numbers balance out better.
  ;; The '10' is picked through trial-and-error. If the target stock is too small, too many consuming agents will switch away from this model due to
stock depletion.
  set a_target-stock-size ceiling (10 * sqrt (abs expected-outflow-per-btu)) + 1  ;; the + 1 is to account for half-used products being counted as
full. the 'abs' is necessary to safeguard for rounding errors (sqrt -0.0000001 gives an error)

  if g_display-stock-size-updates? [ show (word "NEW TARGET STOCK SIZE: " a_target-stock-size "  expected outflow: " expected-outflow-per-btu) ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;                                                          ;;;;;
;;;;;                       Go                                 ;;;;;
;;;;;                                                          ;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
to go ; observer procedure
  if g_stop-now? [ stop ]
  tick

  if g_policy?
  [progress-scenario]

  ;; PBs act first in the model. They periodically reconsider their business model (MM+SM+capacity), as well as their selling prices and target stock
levels. Then they restock up to the target levels.
  ask Producing-Businesses
  [ ;; if it's time for a strategic reconsideration, evaluate options for MM+SM+capacity, as well as the optimal price and stock target levels
    if current-tick = a_time-of-next-strategic-reconsideration
    [ reconsider-production-strategy ]

    ;; if it's time for a tactical reconsideration, optimise price and target stock level
    if current-tick = a_time-of-next-tactical-reconsideration
    [ reconsider-tactical-production-choices ]

    ;; produce up to the target stock levels (insofar capacity allows)
    ask g_active-producers [ produce-output ]
  ]


  ;; CBs/Cs act after the PBs. They attempt to restock products, periodically evaluate their consumption model, and periodically evaluate their
supplier. Then they satisfy their Need through the chosen CM
  ask g_all-consuming-agents
  [ ;; If necessary, try to restock for this round. If the product is no longer available from the supplier, this will trigger a strategic
reconsideration
    if ([a_automatic-repurchase?] of a_Consumption-Model) and (is-agent? a_current-suppliers-SM)
    [ restock-products ]

    ;; if it's time for a strategic reconsideration, evaluate all offers on the market, also for different CMs
    if a_time-of-next-strategic-reconsideration = current-tick
    [ reconsider-consumption-strategy false 0 0 ]

    ;; if it's time for a tactical reconsideration, evaluate all offers on the market for the current CM
    if current-tick = a_time-of-next-tactical-reconsideration
    [ reconsider-tactical-consumption-choices ]

    ;; all CBs/Cs satisfy their need by consuming input products and disposing waste
    satisfy-need
  ]

  ;; recalculate market shares
  update-market-shares

  ;; calculate LCA scores (use of resources and generated wastes)
  update-resource-LCA-scores

  ;; if switched on, plot the new values on the dashboard graphs based on the current situation
  if g_show-plots? [plot-graphs]
end



to progress-scenario ; observer procedure
  ;; this procedure makes sure that Policy Instruments and Market Developments activate (deactivate) when their starting (ending) condition is met

  ask Policy-Instruments-lib
  [ ;; The renting scheme subsidy lowers the variable costs of selling things as a Service on this market
    if a_name = "Subsidy for setting up a renting scheme" and PI-sharing-subsidy?
    [ ifelse current-tick = policy-start-time
      [ set a_active? true
        ask a_activation-icon [ set color green ]
        ask Sales-Models-lib [ if a_main-output-is-Service? [ set a_other-variable-costs-per-unit a_other-variable-costs-per-unit - 1 ] ]
        ask Sales-Models [ if a_main-output-is-Service? [ set a_other-variable-costs-per-unit a_other-variable-costs-per-unit - 1 ] ]
      ]
      [ if current-tick = policy-end-time
        [ set a_active? false
          ask a_activation-icon [ set color red ]
          ask Sales-Models-lib with [ a_main-output-is-Service? ] [ set a_other-variable-costs-per-unit a_other-variable-costs-per-unit + 1 ]
          ask Sales-Models with [ a_main-output-is-Service? ] [ set a_other-variable-costs-per-unit a_other-variable-costs-per-unit + 1  set a_offer-
price-per-unit a_offer-price-per-unit + 1]
        ]
      ]
    ]

    ;; The tax on using Plastic A increases the (variable) manufacturing costs of MMs that have it as input
    if a_name = "Tax on use of Plastic A" and PI-tax-on-Plastic-A?
    [ ifelse current-tick = policy-start-time
      [ set a_active? true
        ask a_activation-icon [ set color green ]
        let plastic-A-ID [a_library-id] of one-of Consumables-lib with [a_name = "Plastic A"]
        ask (turtle-set Manufacturing-Models-lib Manufacturing-Models)
        [ if member? plastic-A-ID a_secondary-input-ids-list
          [ set a_other-variable-costs-per-unit a_other-variable-costs-per-unit + 30 ]
        ]
      ]
      [ if current-tick = policy-end-time
        [ set a_active? false
          ask a_activation-icon [ set color red ]
          let plastic-A-ID [a_library-id] of one-of Consumables-lib with [a_name = "Plastic A"]
          ask (turtle-set Manufacturing-Models-lib Manufacturing-Models)
          [ if member? plastic-A-ID a_secondary-input-ids-list
            [ set a_other-variable-costs-per-unit a_other-variable-costs-per-unit - 30 ]
          ]
        ]
      ]
    ]
```

```
    ;; The environmental awareness campaign increases the weight Consumers put on the environment (up to the maximum value on the 1-5 scale)
    if a_name = "Environmental awareness campaign" and  PI-env-campaign?
    [ ifelse current-tick = policy-start-time
      [ set a_active? true
        ask a_activation-icon [ set color green ]
        ask g_all-consuming-agents
        [ let env-pref item 1 a_preference-weights-list
          set env-pref min (list (env-pref + 2) 5)
          set a_preference-weights-list replace-item 1 a_preference-weights-list env-pref
        ]
      ]
      [ if current-tick = policy-end-time
        [ set a_active? false
          ask a_activation-icon [ set color red ]
        ]
      ]
    ]
  ]
end

to reconsider-production-strategy  ;; PB procedure
  ;; This very layered procedure  let's a PB evaluate all options for configurations of MM, SM-P and SM-S.
  ;; We distinguish three types of configuration changes: (1) just optimise current configuration, (2) same MM but different SM configuration, (3)
different MM and one new SM
  ;; For every configuration option, the PB calculates the optimal price and capacity based on market research
  ;; It then selects and adopts the configuration with the best trade-off between expected profit and expected fit

  ;; Select the sample of (potential) customers that will be used for market research throughout the entire procedure
  set rec_CA-sample select-CA-sample a_market-research-relative-fraction

  ;; initialize some of the temporary global  variables used for PB strategic reconsiderations
  set rec_current-MM-ID ifelse-value (is-agent? a_MM) [ [a_library-ID] of a_MM] [0]
  set rec_current-SM-P-ID ifelse-value (is-agent? a_product-based-SM) [ [a_library-ID] of a_product-based-SM] [0]
  set rec_current-SM-S-ID ifelse-value (is-agent? a_service-based-SM) [ [a_library-ID] of a_service-based-SM] [0]
  set rec_MM-ID rec_current-MM-ID
  set rec_SM-P-ID rec_current-SM-P-ID
  set rec_SM-S-ID rec_current-SM-S-ID

  ;; CAPACITY CHANGES (no SM configuration change. No capacity change equals stay with the current situation)
  reconsider-price-and-target-stock-size a_product-based-SM a_service-based-SM false true rec_CA-sample

  compare-new-strategic-option-with-best true 0

  if g_display-PB-strategic-reconsideration-updates?
  [ show (word "SCORES after capacity reconsideration. MM-ID: " rec_best-MM-ID "  P-ID: "  rec_best-SM-P-ID "  S-ID: " rec_best-SM-S-ID "  capacity,
prices, volumes: " rec_best-capacity " " rec_best-price-P " " rec_best-price-S " " rec_best-volume-P " " rec_best-volume-S "  profit: " rec_best-profit
"  max violation: " rec_best-max-violation "  fit: " rec_best-fit) ]

  ;; SM CHANGES (no capacity change)

  ;; create the overview of possible combinations of P and S, including leaving one empty.
  let MM-SMs-configurations-nlist generate-SMs-switch-options-nlist
  evaluate-MM-SMs-options-for-PB-strategic-change MM-SMs-configurations-nlist

  ;; MM CHANGES (one SM, incl capacity optimisation)
  set MM-SMs-configurations-nlist generate-MM-SMs-switch-options-nlist [a_library-id] of a_MM
  evaluate-MM-SMs-options-for-PB-strategic-change MM-SMs-configurations-nlist

  if g_display-PB-strategic-reconsideration-updates? [ show (word rec_best-MM-ID " " rec_best-SM-P-ID " " rec_best-SM-S-ID) ]

  if g_display-strategic-changes? and ((list rec_current-MM-ID rec_current-SM-P-ID rec_current-SM-S-ID) != (list rec_best-MM-ID rec_best-SM-P-ID
rec_best-SM-S-ID))
  [ show (word "tick " current-tick ". Producing-Business " who " switches from " (list rec_current-MM-ID rec_current-SM-P-ID rec_current-SM-S-ID) " to
" (list rec_best-MM-ID rec_best-SM-P-ID rec_best-SM-S-ID rec_best-capacity " " rec_best-price-P " " rec_best-price-S " " rec_best-volume-P " "
rec_best-volume-S)) ]

  ;; adopt best option
  if rec_best-MM-ID != rec_current-MM-id
  [ remove-MM
    adopt-MM rec_best-MM-ID
    set a_cash-balance a_cash-balance - [a_initial-investment-cost] of library-object-with-id rec_best-MM-ID
  ]

  ;; if any change happens to the product-based SM:
  if rec_best-SM-P-ID != rec_current-SM-P-ID
  [
    remove-SM a_product-based-SM
    let volume-P-per-btu rec_best-volume-P / a_ROI-period
    if rec_best-SM-P-ID != 0
    [ adopt-SM rec_best-SM-P-ID (list 0 rec_best-price-P 0) volume-P-per-btu
      set a_cash-balance a_cash-balance - [a_initial-investment-cost] of library-object-with-id rec_best-SM-P-ID
    ]
  ]

  ;; if any change happens to the service-based SM:
  if rec_best-SM-S-ID != rec_current-SM-S-ID
  [ remove-SM a_service-based-SM
    let volume-S-per-btu rec_best-volume-S / a_ROI-period
    if rec_best-SM-S-ID != 0
    [ adopt-SM rec_best-SM-S-ID (list 0 rec_best-price-S 0) volume-S-per-btu
      set a_cash-balance a_cash-balance - [a_initial-investment-cost] of library-object-with-id rec_best-SM-S-ID
    ]
  ]

  adopt-capacity rec_best-capacity

  ask g_all-consuming-agents [ reconsider-consumption-strategy true nobody nobody ]
```

```
  if is-agent? a_product-based-SM [ ask a_product-based-SM [ set a_total-production-shortage 0 ] ]
  if is-agent? a_service-based-SM [ ask a_service-based-SM [ set a_total-production-shortage 0 ] ]

  set a_time-of-next-strategic-reconsideration current-tick + a_strategic-reconsideration-interval
  set a_time-of-next-tactical-reconsideration current-tick + a_tactical-reconsideration-interval
end

to-report select-CA-sample [ fraction ]  ;; PB procedure
  ;; This procedure samples a group of customers, preferably divided 50-50 among current customers of the PB, and curreny non-customers

  let asking-PB self
  let CA-sample (turtle-set)

  let current-customers g_all-consuming-agents with [ is-agent? a_current-suppliers-SM and (a_current-supplier = asking-PB) ]
  let current-non-customers g_all-consuming-agents with [ (not is-agent? a_current-suppliers-SM) or (a_current-supplier != asking-PB) ]

  let needed-sample-size ceiling (fraction * count g_all-consuming-agents)

  let n-current-customers count current-customers
  let n-current-non-customers count current-non-customers

  let target-n-current-customers-of-each-group (floor (needed-sample-size * 0.5))

  let n-current-customers-in-sample 0
  let n-current-non-customers-in-sample 0

  ifelse n-current-non-customers >= target-n-current-customers-of-each-group
  [ set n-current-customers-in-sample min (list target-n-current-customers-of-each-group n-current-customers)
    set n-current-non-customers-in-sample (2 * target-n-current-customers-of-each-group) - n-current-customers-in-sample
  ]
  [ set n-current-non-customers-in-sample n-current-non-customers
    set n-current-customers-in-sample (2 * target-n-current-customers-of-each-group) - n-current-non-customers-in-sample
  ]

  let current-customers-in-sample n-of n-current-customers-in-sample current-customers
  let current-non-customers-in-sample n-of n-current-non-customers-in-sample current-non-customers

  set CA-sample add-to-turtle-set current-customers-in-sample CA-sample
  set CA-sample add-to-turtle-set current-non-customers-in-sample CA-sample

  set a_fraction-of-current-customers-chosen-for-CA-sample ifelse-value (any? current-customers) [(sum [a_Need-per-btu] of current-customers-in-sample)
/ (sum [a_Need-per-btu] of current-customers) ] [ 0 ]
  set a_fraction-of-current-non-customers-chosen-for-CA-sample ifelse-value (any? current-non-customers) [(sum [a_Need-per-btu] of current-non-
customers-in-sample) / (sum [a_Need-per-btu] of current-non-customers) ] [ 0 ]

  report CA-sample
end

to-report generate-SMs-switch-options-nlist ;; PB procedure
  ;; this procedure returns all possibilities for new SM configurations for this PB, given the current MM.

  let SM-P-IDs (list 0 ([a_associated-product-based-SM-id] of a_MM))
  let SM-S-IDs fput 0 ([a_possible-service-based-SM-ids] of a_MM)

  let MM-id [a_library-id] of a_MM

  let MM-SMs-configurations-nlist (list)

  foreach SM-P-IDs
  [ let P-ID ?
    foreach SM-S-IDs
    [ let S-ID ?

      ;; check that not both IDs are zero, and that it's not the same configuration as the current configuration, because we already considered that
one when considering just a capacity change
      if not (((P-ID = 0) and (S-ID = 0)) or ((P-ID = rec_current-SM-P-ID) and (S-ID = rec_current-SM-S-ID)))
      [ set MM-SMs-configurations-nlist lput (list MM-id P-ID S-ID) MM-SMs-configurations-nlist ]
    ]
  ]
  report MM-SMs-configurations-nlist
end

to-report generate-MM-SMs-switch-options-nlist [ current-MM-ID ] ;; PB procedure
  ;; this procedure returns all possibilities for this PB to choose a different MM with one associated SM

  let available-MMs Manufacturing-Models-lib with [ a_available? ]

  let MM-SMs-configurations-nlist (list)

  ask available-MMs
  [ let MM-id a_library-id
    let SM-P-id a_associated-product-based-SM-id
    set MM-SMs-configurations-nlist lput (list MM-id SM-P-id 0) MM-SMs-configurations-nlist

    let SM-S-IDs a_possible-service-based-SM-ids
    foreach SM-S-ids
    [ let SM-S-id ?
      set MM-SMs-configurations-nlist lput (list MM-id 0 SM-S-id) MM-SMs-configurations-nlist
    ]
  ]

  report MM-SMs-configurations-nlist
end

to evaluate-MM-SMs-options-for-PB-strategic-change [ MM-SMs-configurations-nlist ]
  ;; This procedure evaluates all identified configurations of MM and SMs, calculates max expected profit and weighs it against strategic fit to find
the best overall configuration

  foreach MM-SMs-configurations-nlist
```

```
  [
    if g_display-PB-strategic-reconsideration-updates? [ show (word "New SM combination under consideration: " ?) ]
    set rec_MM-id first ?
    set rec_SM-P-ID item 1 ?
    set rec_SM-S-ID last ?

    let update-P? (rec_SM-P-ID > 0)
    let update-S? (rec_SM-S-ID > 0)

    let changed-MM? (rec_MM-id != rec_current-MM-ID)
    let changed-SM-P? (rec_SM-P-ID != rec_current-SM-P-ID)
    let changed-SM-S? (rec_SM-S-ID != rec_current-SM-S-ID)

    ;; calculate switching costs
    let MM-lib library-object-with-id rec_MM-id
    let SM-P-lib ifelse-value update-P? [library-object-with-id rec_SM-P-ID] [nobody]
    let SM-S-lib ifelse-value update-S? [library-object-with-id rec_SM-S-ID] [nobody]

    let MM a_MM
    let SM-P nobody
    let SM-S nobody

    if changed-MM?
    [ ask MM [ set a_temporarily-disabled? true ]
      set MM generate-ghost-MM MM-lib
    ]

    if update-P?
    [ ifelse changed-SM-P?
      [ if has-product-SM? [ ask a_product-based-SM [ set a_temporarily-disabled? true ] ]
        set SM-P generate-ghost-SM SM-P-lib MM
      ]
      [ set SM-P a_product-based-SM ]
    ]

    if update-S?
    [ ifelse changed-SM-S?
      [ if has-service-SM? [ ask a_service-based-SM [ set a_temporarily-disabled? true ] ]
        set SM-S generate-ghost-SM SM-S-lib MM
      ]
      [ set SM-S a_service-based-SM ]
    ]

    let switching-costs 0
    set switching-costs switching-costs + calculate-total-net-switching-costs a_MM MM-lib false
    if update-P? [ set switching-costs switching-costs + calculate-total-net-switching-costs a_product-based-SM SM-P-lib false ]
    if update-S? [ set switching-costs switching-costs + calculate-total-net-switching-costs a_service-based-SM SM-S-lib false ]

    if g_display-PB-strategic-reconsideration-updates? [ show (word "update-P? " update-P? "  update-S? " update-S? "  SM-P: " SM-P "  SM-S: " SM-S) ]

    reconsider-price-and-target-stock-size SM-P SM-S false true rec_CA-sample

    ;; compare new configuration with best configuration found so far
    compare-new-strategic-option-with-best false switching-costs

    ;; remove temporary SM object instances
    if changed-MM? [ ask MM [ die ] ]
    if update-P? and changed-SM-P? [ ask SM-P [ die ] ]
    if update-S? and changed-SM-S? [ ask SM-S [ die ] ]

    if changed-MM? [ ask a_MM [ set a_temporarily-disabled? false ] ]
    if changed-SM-P? and has-product-SM? [ ask a_product-based-SM [ set a_temporarily-disabled? false ] ]
    if changed-SM-S? and has-service-SM? [ ask a_service-based-SM [ set a_temporarily-disabled? false ] ]
  ]
end

to-report calculate-PB-highest-max-violation [ SM-P SM-S ] ; PB procedure
  ;; This procedure compares the proposed SM-P and SM-S with the thresholds of the PB, and returns the highest relative violation

  let violation-P ifelse-value (is-agent? SM-P) [calculate-max-preference-threshold-violation SM-P false] [0]
  let violation-S ifelse-value (is-agent? SM-S) [calculate-max-preference-threshold-violation SM-S false] [0]
  report max (list violation-P violation-S)
end

to-report calculate-weighted-strategic-fit [ SM-P SM-S allocation-P allocation-S ] ; PB procedure
  ;; This procedure compares the proposed SM-P and SM-S with the strategic Preference weights of the PB, with relative weights based on output volume,
and returns the weighted fit

  let has-P? is-agent? SM-P
  let has-S? is-agent? SM-S

  let fit-P ifelse-value has-P? [ calculate-preference-fit SM-P ] [ 0 ]
  let fit-S ifelse-value has-S? [ calculate-preference-fit SM-S ] [ 0 ]
  let total-allocation allocation-P + allocation-S
  ifelse total-allocation > 0
  [ let relative-weight-P allocation-P / total-allocation
    let relative-weight-S allocation-S / total-allocation
    report fit-P * relative-weight-P + fit-S * relative-weight-S
  ]
  [ ifelse has-P? and has-S?
    [ report fit-P * 0.5 + fit-S * 0.5 ]
    [ ifelse has-P?
      [ report fit-P ]
      [ report fit-S ]
    ]
  ]
end

to-report generate-ghost-MM [ MM-lib ] ; PB procedure
```

```
  ;; This procedure creates a temporary MM, allowing the PB to evaluate it during strategic reconsideration

  let owner self
  let new-MM nobody

  ;; create an SM instance, ensure ownership and visuals
  ask MM-lib
  [ hatch-Manufacturing-Models 1
    [ set new-MM self
      set a_owner owner
      set a_library-object myself
      set a_temporarily-disabled? false
    ]
  ]
  report new-MM
end

to-report generate-ghost-SM [ SM-lib associated-MM ] ; PB procedure
  ;; This procedure creates a temporary SM, allowing the PB to evaluate it during strategic reconsideration

  let owner self
  let new-SM nobody

  ;; create an SM instance, ensure ownership and visuals
  ask SM-lib
  [ hatch-Sales-Models 1
    [ set new-SM self
      set a_owner owner
      set a_associated-MM associated-MM
      set a_library-object myself
      set a_temporarily-disabled? false
    ]
  ]
  report new-SM
end

to compare-new-strategic-option-with-best [ first-option? switching-costs ] ; PB procedure
  ;; procedure to compare a new configuration of MM and SM(s) with the one currently identified as best. The optimal price and associated profit have
already been calculated in a previous step

  ;; calculate preference max violation and fit
  let new-max-violation calculate-PB-highest-max-violation opt_best-volume-P-for-this-config opt_SM-S
  let new-fit calculate-weighted-strategic-fit opt_SM-P opt_SM-S opt_best-volume-P-for-this-config opt_best-volume-S-for-this-config

  ;; subtract switching cost from best found profit
  let new-profit opt_best-profit-for-this-config - switching-costs

  let both-options-make-profit? (rec_best-profit > 0) and (new-profit > 0)
  let new-configuration-is-best? first-option?

  ifelse first-option?
  [

  ]
  [ ;; FIRST COMPARISON RULE
    ;; Making any profit is a hard constraint: if both configurations cause a loss, the configuration with the lowest budget violation is always chosen
as best configuration.
    ;; If both configurations lead to some profit,  the procedure will continue to a further comparison that also takes into account the agent's
Preferences
    if not both-options-make-profit? [ set new-configuration-is-best? new-profit > rec_best-profit ]

    ;; SECOND COMPARISON RULE: preference threshold violation
    if both-options-make-profit?
    [ ;; evaluate SM threshold violations of Preferences
      let any-preference-threshold-violation? (rec_best-max-violation > 0) or (new-max-violation > 0)
      ifelse any-preference-threshold-violation?
      [ set new-configuration-is-best? new-max-violation < rec_best-max-violation ]

      ;; THIRD COMPARISON RULE
      [ ;; if this is not the first configuration found that respects all thresholds, compare it with the best configuration found so far by weighing
money aspects against non-money aspects (i.e. lifestyle/strategic fit and loyalty score)
        let fit-difference new-fit - rec_best-fit

        ;; compare the values for the 'current best configuration' (for the CM now under consideration) with the values for the overall best
configuration found so far
        let profit-difference-per-btu new-profit - rec_best-profit
        let highest-possible-profit max (list rec_best-profit new-profit)

        if g_display-PB-strategic-reconsideration-updates? [ show (word " prof diff: " profit-difference-per-btu "  fit diff: " fit-difference "  rel
wtp for pref fit: " a_relative-wtp-for-preference-fit "  highest profit: " highest-possible-profit) ]

        ;; calculate the monetary equivalent of the better (or worse) strategic fit, relative to the highest possible profit, and compare it with the
difference in profit
        if profit-difference-per-btu + fit-difference * a_relative-wtp-for-preference-fit * highest-possible-profit > 0
        [ set new-configuration-is-best? true ]
      ]
    ]

    if g_display-PB-strategic-reconsideration-updates?
    [ show ""
      show (word "OLD SCORES. MM-ID: " rec_best-MM-ID "  P-ID: "  rec_best-SM-P-ID "  S-ID: " rec_best-SM-S-ID "  capacity, prices, volumes: "
rec_best-capacity " " rec_best-price-P " " rec_best-price-S " " rec_best-volume-P " " rec_best-volume-S "  profit: " rec_best-profit "  max violation:
" rec_best-max-violation "  fit: " rec_best-fit)
      show (word "NEW SCORES. MM-ID: " rec_MM-ID "  P-ID: "  rec_SM-P-ID "  S-ID: " rec_SM-S-ID "  capacity, prices, volumes: " opt_best-capacity-for-
this-config " " opt_best-price-P-for-this-config " " opt_best-price-S-for-this-config " " opt_best-volume-P-for-this-config " " opt_best-volume-S-for-
this-config "  profit: " new-profit "  max violation: " new-max-violation "  fit: " new-fit)
      show ifelse-value new-configuration-is-best? [ "new configuration is best!" ] [ "old configuration was best!" ]
    ]
  ]
```

```
  if new-configuration-is-best?
  [ set rec_best-MM-ID rec_MM-ID
    set rec_best-SM-P-ID rec_SM-P-ID
    set rec_best-SM-S-ID rec_SM-S-ID
    set rec_best-capacity opt_best-capacity-for-this-config
    set rec_best-price-P opt_best-price-P-for-this-config
    set rec_best-price-S opt_best-price-S-for-this-config
    set rec_best-volume-P opt_best-volume-P-for-this-config
    set rec_best-volume-S opt_best-volume-S-for-this-config
    set rec_best-net-profit-P opt_best-net-profit-P-for-this-config
    set rec_best-net-profit-S opt_best-net-profit-S-for-this-config
    set rec_best-profit new-profit
    set rec_best-max-violation new-max-violation
    set rec_best-fit new-fit
  ]
end

to reconsider-tactical-production-choices ;; PB procedure
  ;; during this procedure, the PB reconsiders price and target stock levels of its SM(s)

  ;; first, schedule the time of next tactical reconsideration
  set a_time-of-next-tactical-reconsideration current-tick + a_tactical-reconsideration-interval

  ;; then do the actual reconsideration
  let CA-sample select-CA-sample a_market-research-relative-fraction
  reconsider-price-and-target-stock-size a_product-based-SM a_service-based-SM true false CA-sample
end

to initialise-stocks [ for-input? for-output? ] ; MM/SM procedure
  ;; This procedure creates (empty) initial stocks for new MMs and SMs. This initialisation is required to prevent errors

  if for-input?
  [ ifelse a_main-input-is-Consumable?
    [ set a_main-input-stock (turtle-set make-initial-empty-Consumable true) ]
    [ set a_main-input-stock (turtle-set) ]
  ]

  if for-output?
  [ ifelse a_main-output-is-Consumable?
    [ set a_main-output-stock (turtle-set make-initial-empty-Consumable false) ]
    [ set a_main-output-stock (turtle-set) ]
  ]
end

to adopt-MM [ MM-id ]  ;; PB procedure
  ;; procedure to adopt a new MM by acquiring required skills and infrastructures, creating the new model, linking it to the PB and visualising it

  let new-MM nobody
  let owner self

  acquire-needed-skills-and-infrastructures-for-new-model MM-id

  ;; create an MM instance, ensure ownership and visuals
  ask library-object-with-id MM-id
  [ hatch-Manufacturing-Models 1
    [ set a_library-object myself
      set new-MM self
      set a_owner owner
      set a_temporarily-disabled? false

      ask a_owner [ set a_MM myself ]
      visualise-model

      initialise-stocks false true
    ]
  ]
end

to remove-MM ;; PB procedure
  ;; procedure to remove the current MM (needed before adopting a new one)

  let root-agent self
  if is-agent? a_MM
  [ ;; dispose of remaining stocks and update the cash balance accordingly
    ask a_MM
    [ ask a_main-output-stock
      [ let disposal-cost calculate-cost-of-product-disposal a_library-id remaining-amount
        ask root-agent [ set a_cash-balance a_cash-balance - disposal-cost ]
        die  ;; the products disappear from the model
      ]
      die  ;; the SM disappears from the model
    ]
  ]
end

to adopt-SM [ SM-id prices-list expected-sales-per-btu ]  ;; PB procedure
  ;; procedure to adopt a new SM by acquiring required skills and infrastructures, creating the new model, linking it to the PB and the MM and
visualising it

  let new-SM nobody
  let owner self

  acquire-needed-skills-and-infrastructures-for-new-model SM-id

  ;; create an SM instance, ensure ownership and visuals
  ask library-object-with-id SM-id
  [ hatch-Sales-Models 1
    [ if a_main-output-is-Service? = 0 [ inspect myself error "" ]
```

```
      set new-SM self
      set a_owner owner
      set a_associated-MM [a_MM] of a_owner
      set a_temporarily-disabled? false
      visualise-model

      ask a_owner
      [ ifelse [a_main-output-is-Service?] of new-SM
        [ set a_service-based-SM new-SM ]
        [ set a_product-based-SM new-SM ]
      ]

      set a_library-object myself

      initialise-stocks true true

      set-SM-price self prices-list
      ifelse a_main-output-is-Tool?
      [ update-target-stock-size expected-sales-per-btu ]
      [ update-target-stock-size 0 ]
    ]
  ]

  set-color-PB
end

to remove-SM [ SM-to-remove ] ; PB procedure
  ;; if a PB changes to a new SM, she first removes her current SM in that slot (either product- or service-based), meaning that it disappears from the
model and remaining stocks are disposed of (either to WM or PE)

  ;; ask all current customers to schedule a reconsideration for this tick
  ask g_all-consuming-agents
  [ if (a_current-suppliers-SM = SM-to-remove) and ([a_automatic-repurchase?] of a_Consumption-Model) [ set a_time-of-next-strategic-reconsideration
current-tick ]
    if a_best-offer-SM = SM-to-remove [ set a_best-offer-SM nobody ]
  ]

  let root-agent self
  if is-agent? SM-to-remove
  [ ;; dispose of remaining stocks and update the cash balance accordingly
    ask SM-to-remove
    [ let relevant-stocks (turtle-set a_main-input-stock a_main-output-stock)

      ask relevant-stocks
      [ let disposal-cost calculate-cost-of-product-disposal a_library-id remaining-amount
        ask root-agent [ set a_cash-balance a_cash-balance - disposal-cost ]
        die  ;; the products disappear from the model
      ]
      die  ;; the SM disappears from the model
    ]
  ]
  set-color-PB
end

to reconsider-tactical-consumption-choices  ;; CB/C procedure
  ;; procedure to evaluate current offers on the market that correspond with the current CM

  ;; first, schedule the time of next tactical reconsideration
  set a_time-of-next-tactical-reconsideration current-tick + a_tactical-reconsideration-interval

  ;; find the best offer on the market for the current CM
  find-best-input-offer-for-given-CM a_Consumption-Model false true 0 0  ;; 'false' indicates that the procedure is not called in the context of
market-research. 'true' indicates that only the tactical reconsideration cycle should be taken into account here

  ifelse is-agent? a_best-offer-SM
  ;; if a suitable offer has been found, check if it is from a different supplier than the current offer. If so, make the switch
  [ if a_current-suppliers-SM != a_best-offer-SM [ switch-to-best-offer ] ]
  ;; if the agent was unable to find any suitable offer for her CM, she will directly perform a strategic reconsideration to change her CM
  [ set a_time-of-next-strategic-reconsideration current-tick ]

  ; if it's not during the initialisation and products are not automatically repurchased, find out how much product to buy
  if current-tick > 0 and [not a_automatic-repurchase?] of a_Consumption-Model
  [ update-needed-n-new-products ]

  ;; regardless of whether the CM has changed, the agent should restock to make sure she can satisfy her Need this tick
  restock-products
end

to find-best-input-offer-for-given-CM [consumption-model for-market-research? tactical? forbidden-SM-1 forbidden-SM-2]   ;; CB/C procedure
  ;; finds the best input contract based on overall costs per btu and loyalty.

  let current-CM a_Consumption-Model
  let current-SM a_current-suppliers-SM

  ;; find offers on the market that have sufficient stocks
  let suitable-offers-on-market [suitable-Sales-Models-for-input-id-and-Need a_main-input-id ([a_Need-per-btu] of myself) current-CM current-SM
forbidden-SM-1 forbidden-SM-2] of consumption-model

  ;; calculate preference violation and fit for the CM (store the values under a_best-offer-xxx)
  update-best-offer-preference-violation-and-fit consumption-model

  if any? suitable-offers-on-market
  [ set a_best-offer-SM -1  ; -1 signals that the current best offer needs to be reset through a fresh round of comparison of offers on the market
    let offers-list sort-on [random 10] suitable-offers-on-market  ;; this line is just to convert a turtle-set into a list, the sorting is irrelevant

    ;; find the best offer by iterating through all offers found above
    foreach offers-list
    [ compare-with-best-offer-for-same-CM consumption-model ? for-market-research? tactical? ]
  ]
```

```
end

to compare-with-best-offer-for-same-CM [ consumption-model offer for-market-research? tactical? ]  ;; CB/C procedure
  ;; Compares the specified 'offer' (refers to an SM) with the currently identified best offer ('a_best-offer') based on total costs per btu and
loyalty.

  ;; calculate costs and (possibly) profit
  let total-costs-per-btu calculate-total-costs-per-btu-for-CM consumption-model offer for-market-research? (not tactical?) true
  let total-profit-per-btu ifelse-value (breed = Consuming-Businesses) [ a_default-revenue-per-btu * [a_revenue-multiplier] of consumption-model -
total-costs-per-btu  - a_default-cost-per-btu ] [ 0 ]

  ;; calculate virtual loyalty bonus (i.e. discount to the calculated price)
  let virtual-loyalty-bonus-new-offer calculate-virtual-loyalty-bonus-for-offer offer total-costs-per-btu total-profit-per-btu

  ;; make the final comparison based on total cost and virtual loyalty bonus for both offers. If the new one is better, update 'a_best-offer-...'
attributes
  if (a_best-offer-SM = -1) or (total-costs-per-btu - virtual-loyalty-bonus-new-offer < a_best-offer-costs-per-btu - a_best-offer-virtual-loyalty-
bonus)
  [ update-best-offer-attributes offer total-costs-per-btu virtual-loyalty-bonus-new-offer total-profit-per-btu a_best-offer-fit a_best-offer-
preference-violation ]
end

to-report calculate-total-costs-per-btu-for-CM [ CM-lib offer for-market-research? include-offer-price? include-switching-costs? ] ; CB/C procedure
  ;; procedure to calculate total costs per btu over a certain period, taking into account all offer price aspects as well as switching costs

  let switching-costs-per-btu 0
  if include-switching-costs?
  [ let total-switching-costs calculate-total-net-switching-costs a_Consumption-Model CM-lib for-market-research?
    ;; add to the best offer costs the switching costs (calculated per btu based on the ROI period)
    set switching-costs-per-btu total-switching-costs / a_ROI-period
  ]

  let consideration-period determine-applicable-consideration-period-CA CM-lib for-market-research?

  ;; sum all cost components, all specified per btu
  let offer-based-costs-per-btu 0
  if include-offer-price?
  [ ;; take into account that if you consider a new offer for your CURRENT consumption model, your currently owned product(s) may reduce the number of
products needed to be bought over the upcoming period
    let required-new-n-products-in-period calculate-total-required-new-n-products-in-period for-market-research? CM-lib consideration-period
    let required-new-n-products-per-btu required-new-n-products-in-period / consideration-period
    set offer-based-costs-per-btu ([a_offer-price-per-unit] of offer) * required-new-n-products-per-btu
  ]
  ;; determine (variable) costs except for main input purchasing: costs of buying secondary inputs from the World Market and of disposing secondary
outputs (or revenues of selling them to WM), and other variable costs
  let other-unit-based-costs-per-btu [calculate-cost-of-secondary-inputs-and-outputs + a_other-variable-costs-per-unit] of CM-lib * a_Need-per-btu

  let one-time-costs-per-btu ([a_offer-price-one-time-contract-cost] of offer) / consideration-period
  let time-based-costs-per-btu [a_offer-price-per-btu] of offer + [a_structural-costs-per-btu] of CM-lib

  report switching-costs-per-btu + offer-based-costs-per-btu + other-unit-based-costs-per-btu + one-time-costs-per-btu + time-based-costs-per-btu
end

to-report determine-applicable-consideration-period-CA [ relevant-CM-or-CM-lib  for-market-research? ]  ;; CB/C procedure
  ;; procedure to determine the consideration period that the Consumer will use to convert one-time costs to costs per btu

  let consideration-period a_ROI-period
  if not [a_automatic-repurchase?] of relevant-CM-or-CM-lib
  [ ifelse for-market-research?
    [ set consideration-period calculate-product-lifetime relevant-CM-or-CM-lib ]
    [ if (relevant-CM-or-CM-lib = a_Consumption-Model)
      [ let time-until-product-EOL calculate-time-until-product-EOL
        if time-until-product-EOL > 0 [ set consideration-period calculate-time-until-product-EOL ]
      ]
    ]
  ]
  report consideration-period
end

to-report calculate-product-lifetime [ CM-lib ] ;; CB/C procedure
  ;; simple procedure to calculate the total lifetime of a main input product of the specified CM-lib

  let product-id [a_main-input-id] of CM-lib
  let product-total-use-time [a_total-use-time] of (library-object-with-id product-id)
  let use-time-consumed-per-btu a_Need-per-btu * [a_main-input-ratio] of CM-lib
  report floor (product-total-use-time / use-time-consumed-per-btu)
end

to compare-new-offer-on-market-with-best-offer [ new-offer ]  ;; CB/C procedure
  ;; Offers are compared according to structured hierarchical rules: first, always choose the one with the lowest budget violation
  ;; If there are multiple options within the budget, always choose the one with the lowest Preference threshold violation
  ;; If there are multiple options within all Preference thresholds, best combination of cost (or profit) and lifestyle/strategic fit, based on the
willingness to pay

  let current-best-offer a_best-offer-SM
  let possible-new-CM-lib one-of Consumption-Models-lib with [ a_main-input-id = [a_main-output-id] of new-offer ]

  let consideration-period determine-applicable-consideration-period-CA possible-new-CM-lib true

  ;; determine time of potential switch
  let time-until-potential-switch applicable-time-until-next-reconsideration-for-market-research

  ;; calculate costs per btu of new offer
  let new-offer-total-cost-per-btu calculate-total-costs-per-btu-for-CM possible-new-CM-lib new-offer true true true
  let new-offer-profit-per-btu ifelse-value (breed = Consuming-Businesses) [ a_default-revenue-per-btu * [a_revenue-multiplier] of possible-new-CM-lib
- new-offer-total-cost-per-btu - a_default-cost-per-btu ] [ 0 ]

  ;; if so, how much lower should another supplier's offer be before I switch? We take this into account here as a virtual bonus (i.e. discount to the
calculated price)
```

```
  let new-offer-virtual-loyalty-bonus calculate-virtual-loyalty-bonus-for-offer new-offer new-offer-total-cost-per-btu new-offer-profit-per-btu

  let new-offer-max-Preference-threshold-violation calculate-max-preference-threshold-violation possible-new-CM-lib true
  let new-offer-preference-fit calculate-preference-fit possible-new-CM-lib

  update-best-offer-preference-violation-and-fit a_best-offer-SM

  compare-with-best-offer-generic new-offer new-offer-total-cost-per-btu new-offer-profit-per-btu new-offer-max-Preference-threshold-violation new-
offer-preference-fit new-offer-virtual-loyalty-bonus
end

to update-best-offer-preference-violation-and-fit [ offer ] ;; CB/C procedure
  ;; calculate lifestyle/strategic threshold violation
  set a_best-offer-preference-violation calculate-max-preference-threshold-violation offer true

  ;; calculate lifestyle/strategic fit
  set a_best-offer-fit calculate-preference-fit offer
end

to-report calculate-total-required-new-n-products-in-period [ for-market-research? CM-lib consideration-period ] ;; CB/C procedure
  ;; procedure to calculate the number of products that the CB/C will have to buy if it uses the specified CM-lib, given its Need per btu

  ;; the remaining product amount is either specified in number of Tools (e.g. 0.43 if there is one Tool with 43% remaining use time) or Consumable
quantity
  let remaining-matching-product-amount 0

  ;; in case of market research, we ignore currently owned products, because we want to compare the long-term price-per-btu there.
  ;; If not for market research, the consumer is actively reconsidering her consumption strategy and should take her currently owned products during
the comparison.
  if not for-market-research? and (is-agent? a_Consumption-Model and ([a_main-input-id] of CM-lib = [a_main-input-id] of a_Consumption-Model))  ;;
check if the specified CM-lib (associated with the offer) is the same as the current CM
  [ ask a_Consumption-Model
    [ if not a_main-input-is-Service?
      [ set remaining-matching-product-amount sum [remaining-amount] of a_associated-owned-products ]
    ]
  ]

  ;; calculate how many additional products (of the main input) need to be bought in the relevant period
  let consumed-main-input-volume-per-btu a_Need-per-btu * [a_main-input-ratio-in-full-unit-equivalents] of CM-lib
  let required-new-n-products-in-period (consumed-main-input-volume-per-btu * consideration-period - remaining-matching-product-amount)
  if required-new-n-products-in-period < 0 [ set required-new-n-products-in-period 0 ]

  ;show (word who "period: " consideration-period "  remaining amount: " remaining-product-amount "  consumed-main-input-volume-per-btu: " consumed-
main-input-volume-per-btu "  required-new-n-products-per-btu: " required-new-n-products-per-btu)

  report required-new-n-products-in-period
end

to-report calculate-cost-of-secondary-inputs-and-outputs  ;; MM/SM/CM(-lib) procedure
  ;; This procedure returns for any type of model the cost of secondary in/outputs PER UNIT OF MAIN OUTPUT (which is the Need in case of a CM)

  ;; calculates the total current costs of obtaining all secondary inputs, and disposing all secondary outputs, of a model, relative to a unit of main
output (=Need in case of CM)
  let costs-of-secondary-inputs 0

  foreach list-indices a_secondary-input-ids-list
  [ let input-id item ? a_secondary-input-ids-list
    let WM-price-per-unit World-Market-offered-price input-id
    let input-ratio-per-unit-of-output item ? a_secondary-input-ratios-list
    set costs-of-secondary-inputs costs-of-secondary-inputs + input-ratio-per-unit-of-output * WM-price-per-unit
  ]

  let costs-of-secondary-outputs 0
  foreach list-indices a_secondary-output-ids-list
  [ let output-id item ? a_secondary-output-ids-list
    let disposal-price-per-unit calculate-cost-of-product-disposal output-id 1
    let output-amount-per-unit-of-output item ? a_secondary-output-ratios-list
    set costs-of-secondary-outputs costs-of-secondary-outputs + output-amount-per-unit-of-output * disposal-price-per-unit
  ]

  report costs-of-secondary-inputs + costs-of-secondary-outputs
end

to-report applicable-time-until-next-reconsideration-for-market-research ;; CB/C procedure
  ;; this procedure determines how much time there will be before the next strategic reconsideration. Is an input for market research procedures

  let strategic-consideration-period ifelse-value (a_time-of-next-strategic-reconsideration = current-tick) [a_strategic-reconsideration-interval]
[(a_time-of-next-strategic-reconsideration - current-tick)]
  let applicable-time strategic-consideration-period

  if (is-agent? a_Consumption-Model) and [not a_automatic-repurchase?] of a_Consumption-Model
  [ set applicable-time min (list strategic-consideration-period calculate-time-until-product-EOL) ]

  report applicable-time
end

to-report calculate-time-until-product-EOL  ;; CB/C procedure
  ;; short procedure to calculate the time until the associated owned products of the CB/C's current CM are at the end of their functional lifetime

  let consumed-main-input-volume-per-btu a_Need-per-btu * [a_main-input-ratio-in-full-unit-equivalents] of a_Consumption-Model
  let time-until-EOL floor ((sum [ remaining-amount ] of [ a_associated-owned-products ] of a_Consumption-Model) / consumed-main-input-volume-per-btu)
;; the expected lifetime of the associated tool can be calculated based on the consumed input volume per btu (specified in number of Tools, e.g.
0.02343)
  report time-until-EOL
end

to reconsider-consumption-strategy [ for-market-research? forbidden-SM-1 forbidden-SM-2 ] ;; CB/C procedure
  ;; in this procedure, the CB/C evaluates all offers on the market, even those that require a CM switch
```

```
  ;; filter on whether the agent has potential access to all required infrastructures
  let allowed-CMs filter-on-infrastructure-availability Consumption-Models-lib

  ;; filter on CMs for which there is any SM with sufficient stocks. If there is no SM that can supply the required inputs, it is pointless to consider
the respective CM at this point
  let current-CM a_Consumption-Model
  let current-SM a_current-suppliers-SM

  set allowed-CMs allowed-CMs with [ any? suitable-Sales-Models-for-input-id-and-Need a_main-input-id ([a_Need-per-btu] of myself) current-CM current-
SM forbidden-SM-1 forbidden-SM-2 ]

  ifelse count allowed-CMs > 0
  [ ;; now select the best out of the allowed CM's and adopt the best one
    select-best-CM-from-set allowed-CMs forbidden-SM-1 forbidden-SM-2 for-market-research?
  ]
  [ if not for-market-research? [ show "Warning! There is no suitable CM at all for this consumer. This probably means all SMs are either out of stock
or producing at full capacity." ] ]

  if not for-market-research?
  [ ;; if the identified best offer differs from the current CM, make the switch
    if a_current-suppliers-SM != a_best-offer-SM [ switch-to-best-offer ]

    set a_current-offer-total-costs-per-btu a_best-offer-costs-per-btu
    update-best-offer-preference-violation-and-fit a_current-suppliers-SM

    set a_current-offer-fit a_best-offer-fit

    ; if it's not during the initialisation and products are not automatically repurchased, find out how much product to buy
    if current-tick > 0 and [not a_automatic-repurchase?] of a_Consumption-Model
    [ update-needed-n-new-products ]

    ;; regardless of whether the CM has changed, the agent should restock to make sure she can satisfy her Need this tick
    restock-products

    ;; schedule the time of next strategic reconsideration
    set a_time-of-next-strategic-reconsideration current-tick + a_strategic-reconsideration-interval
    ifelse (current-tick + a_tactical-reconsideration-interval < a_time-of-next-strategic-reconsideration)
    [ set a_time-of-next-tactical-reconsideration current-tick + a_tactical-reconsideration-interval ] ;; only use the specified interval if there's
still a full tactical period between now and the next strategic reconsideration
    [ set a_time-of-next-tactical-reconsideration a_time-of-next-strategic-reconsideration ]  ;; otherwise, just set the time of next tactical
reconsideration the same as the next strategic reconsideration

    ;; update current best offer properties for comparing with new offers during future market research
    set a_best-offer-SM -1  ;; this is to make sure 'compare-best-offer-for-CM' will update the attributes based on the newly adopted offer
    compare-with-best-offer-for-same-CM a_Consumption-Model a_current-suppliers-SM true false
  ]
end

to update-needed-n-new-products  ;; CB/C procedure
  ;; determine the number of products the CB/C needs to buy this tick, based on her Need per btu and her currently owned products

  set a_needed-n-new-products ceiling max (list 0 [a_consumed-main-input-volume-per-btu - sum [remaining-amount] of a_associated-owned-products] of
a_Consumption-Model)
end

to-report suitable-Sales-Models-for-input-id-and-Need [ input-id  Need-per-btu current-CM current-SM forbidden-SM-1 forbidden-SM-2 ] ;; CM/CM-lib
procedure
  ;; determine which Sales Models on the market offer the main input for this CM(-lib), and still have some stocks or surplus capacity. Ignore the
specified forbidden-SMs (those are the ones being reconsidered)

  let needed-CM-input-per-btu Need-per-btu * a_main-input-ratio-in-full-unit-equivalents

  ;; determine the minimum stock needed to directly sell the needed initial amount to the buyer
  let total-remaining-amount 0
  if is-agent? current-CM and (input-id = [a_main-input-id] of current-CM)
  [ set total-remaining-amount sum [remaining-amount] of [a_associated-owned-products] of current-CM ]
  let needed-purchase-volume ceiling (needed-CM-input-per-btu - total-remaining-amount)

  ;; perform the checks
  report Sales-Models with [ not (self = forbidden-SM-1) and not (self = forbidden-SM-2) and (not a_temporarily-disabled?) and (a_main-output-id =
input-id) and (sufficient-stock? needed-purchase-volume) and (sufficient-capacity? needed-CM-input-per-btu)]  ;; if it's for market research, we want
to exclude the SM currently selected as best offer
 end

to-report sufficient-stock? [needed-amount] ;; SM procedure
  ;; this procedure reports whether the current stocks of the SM that calls the procedure are sufficient to supply the specified 'needed-amount'

  let sufficient? true
  if current-tick > 0   ;; this check is irrelevant during initialisation
  [ let available-volume ifelse-value (a_main-output-is-Service?) [(stock-size a_main-input-stock) / a_main-input-ratio-in-full-unit-equivalents ]
[stock-size a_main-output-stock]
    set sufficient? available-volume >= needed-amount
  ]
  report sufficient?
end

to-report sufficient-capacity? [ requested-output-volume ] ;; SM procedure
  ;; for offers that involve a transaction at every btu (i.e. automatic repurchase is true for the associated CM), such as Services and certain
Consumables, check if the capacity would allow to provide the additional requested volume specified

  let sufficient? true

  let associated-CM-lib one-of Consumption-Models-lib with [ a_main-input-id = [a_main-output-id] of myself ]

  if [a_automatic-repurchase?] of associated-CM-lib
  [ let needed-input-volume requested-output-volume * a_main-input-ratio-in-full-unit-equivalents
    set sufficient? ([a_output-capacity-per-btu - a_contracted-output-volume-per-btu] of a_associated-MM) >= needed-input-volume
  ]
```

```
  report sufficient?
end


to-report market-research-switch-price-and-volume [ asking-PB offered-SM-or-SM-lib offer-price-list PB-horizon ]  ;; CB/C procedure
  ;; In this procedure, the CB/C determines what price (if any) she is willing to pay for a spefified offer (offered-SM-or-SM-lib), based on
comparisons with other offers on price, thrsholds and preference fit
  ;; offer-price-list is a list with 3 items: one-time-fee (only if Service), unit-based price (always) and time-based fee (only if Service)
  ;; PB-horizon is the time period that the PB looks ahead during his strategic or tactical reconsideration period

  ;; find the corresponding CM-lib
  let main-input-id [a_main-output-id] of offered-SM-or-SM-lib
  let new-CM-lib one-of Consumption-Models-lib with [ a_main-input-id = main-input-id ]
  let output-list-if-agent-will-buy-nothing (list 0 0 a_best-offer-SM)

  ;; determine time of potential switch
  let time-until-potential-switch applicable-time-until-next-reconsideration-for-market-research

  ;; if time until potential switch is longer than the horizon, stop here: the agent will not buy anything within the horizon. The procedure stops
after the 'report'
  if PB-horizon <= time-until-potential-switch [ report output-list-if-agent-will-buy-nothing ]

  ;; calculate remaining consumption period
  let period-of-buying-the-offer (PB-horizon - time-until-potential-switch)

  ; calculate total number of units bought during the period
  let number-of-offered-units-bought-in-PB-period calculate-total-required-new-n-products-in-period true new-CM-lib period-of-buying-the-offer

  if number-of-offered-units-bought-in-PB-period = 0 [ report output-list-if-agent-will-buy-nothing ]

  ;; determine Preference threshold violations
  let max-Preference-threshold-violation calculate-max-preference-threshold-violation new-CM-lib true

  ;; if the current best offer has no budget violation (or if CB: doesn't cause a loss) and a lower max pref threshold violation than the new one, stop
here and report 0
  ifelse breed = Consumers
  [ if (a_best-offer-costs-per-btu <= a_max-cost-threshold-for-unit-of-Need) and (a_best-offer-preference-violation < max-Preference-threshold-
violation) [ report output-list-if-agent-will-buy-nothing ] ]
  [ if (a_best-offer-profit-per-btu >= 0) and (a_best-offer-preference-violation < max-Preference-threshold-violation) [ report output-list-if-agent-
will-buy-nothing ] ]

  ;; determine Preference fit based on current market penetration of the offered product/service
  let preference-fit calculate-preference-fit new-CM-lib

  let fit-difference preference-fit - a_best-offer-fit
  let effective-relative-wtp-for-fit-difference fit-difference * a_relative-wtp-for-preference-fit

  ;; determine loyalty
  let effective-relative-wtp-for-loyalty calculate-effective-relative-wtp-for-loyalty asking-PB

  ;; calculate the break-even-costs: the costs per btu at which both offers (i.e. current best and new) are equally attractive
  let break-even-cost-per-btu calculate-break-even-costs-per-btu new-CM-lib effective-relative-wtp-for-loyalty effective-relative-wtp-for-fit-
difference max-Preference-threshold-violation

  ;; calculate implied offered price: calculate all other relevant costs and then derive the max offer price per unit where the agent will switch
  let consideration-period determine-applicable-consideration-period-CA new-CM-lib true

  let total-independent-costs-per-btu calculate-total-costs-per-btu-for-CM new-CM-lib offered-SM-or-SM-lib true false false

  let number-of-offered-units-bought-in-consideration-period calculate-total-required-new-n-products-in-period true new-CM-lib consideration-period
  let number-of-offered-units-bought-per-btu number-of-offered-units-bought-in-consideration-period / consideration-period

  let max-offer-cost-per-unit (break-even-cost-per-btu - total-independent-costs-per-btu) / number-of-offered-units-bought-per-btu

  if display-additional-sim-opt-updates? or (g_display-market-research-numbers? or g_display-extreme-market-research-results? and ((g_display-CB-
actions? and  (breed = Consuming-Businesses)) or (max-offer-cost-per-unit > 1500) or (g_display-all-relevant-reconsideration-numbers-of-one-Consumer?
and (who = 139))))
  [ show (word "tick " current-tick ": Market research results. Time until next reconsideration: " time-until-potential-switch "  Period of buying the
offer: " period-of-buying-the-offer "  current CM: " [a_name] of a_Consumption-Model " " [who] of a_current-supplier "  current best offer name: "
[a_name] of a_best-offer-SM " " [who] of [a_owner] of a_best-offer-SM "  new offer: " [a_name] of offered-SM-or-SM-lib " " [who] of asking-PB "  best
offer violation: " a_best-offer-preference-violation "  new offer violation: " max-Preference-threshold-violation)
    show (word "                current unit cost: "  [a_offer-price-per-unit] of a_current-suppliers-SM "  current best offer unit cost: "  [a_offer-
price-per-unit] of a_best-offer-SM "  current best offer costs per btu: "  a_best-offer-costs-per-btu "  new offer break-even cost: " break-even-cost-
per-btu "  loyalty-bonus-of-best-offer: " a_best-offer-virtual-loyalty-bonus "  max offer unit costs: " max-offer-cost-per-unit)
    show (word "               a_best-offer-virtual-loyalty-bonus: " a_best-offer-virtual-loyalty-bonus "  effective wtp for loyalty:  " effective-
relative-wtp-for-loyalty "  effective-relative-wtp-for-fit-difference: " effective-relative-wtp-for-fit-difference "  number of products to be bought:
" number-of-offered-units-bought-in-PB-period "  remaining stock: " sum [ remaining-amount ] of [a_associated-owned-products] of a_Consumption-Model  "
consideration-period: " consideration-period "  average number of products used per btu in consideration period: " number-of-offered-units-bought-per-
btu)
  ]

  let volume-of-previous-MR a_last-market-research-volume
  set a_last-market-research-volume number-of-offered-units-bought-in-PB-period

  let is-current-customer? ((is-agent? a_current-suppliers-SM) and (asking-PB = a_current-supplier))

  report (list max-offer-cost-per-unit number-of-offered-units-bought-in-PB-period is-current-customer? a_best-offer-SM a_intermediate-best-offer-SM-
during-MR volume-of-previous-MR offered-SM-or-SM-lib) ; additional debug info: "|" who a_Need-per-btu period-of-buying-the-offer)
end

to-report calculate-virtual-loyalty-bonus-for-offer [ offer offer-costs-per-btu offer-profit-per-btu] ; CB/C procedure
  ;; calculates the absolute amount of money the CB/C is willing to pay on top of the price (when comparing with other offers), simply because it is
offered by her current supplier

  let effective-relative-wtp-for-loyalty calculate-effective-relative-wtp-for-loyalty [a_owner] of offer
  report effective-relative-wtp-for-loyalty * ifelse-value (breed = Consuming-Businesses) [offer-profit-per-btu] [offer-costs-per-btu]
end

to-report calculate-effective-relative-wtp-for-loyalty [ PB-under-consideration ] ; CB/C procedure
```

```
  ;; calculates the relative (to the price) amount of money the CB/C is willing to pay on top of the price (when comparing with other offers), simply
because it is offered by her current supplier

  let same-supplier-for-new-offer? ifelse-value (is-agent? a_current-suppliers-SM) [PB-under-consideration = a_current-supplier] [false]
  report ifelse-value same-supplier-for-new-offer? [ a_relative-wtp-for-loyalty ] [ 0 ]
end

to-report calculate-break-even-costs-per-btu [ new-CM-lib effective-relative-wtp-for-loyalty effective-relative-wtp-for-fit-difference max-Preference-
threshold-violation ] ;; CB/C procedure
  ;; calculate the price at which the CB/C will prefer the specified CM(-lib) over the offer she currently identified as the best

  let break-even-costs-per-btu 0
  ;; if the current best offer violates the cost threshold, choose the new offer whenever total costs are lower
  ifelse breed = Consumers and (a_best-offer-costs-per-btu / a_Need-per-btu > a_max-cost-threshold-for-unit-of-Need)
  [ set break-even-costs-per-btu a_best-offer-costs-per-btu ]  ;; in this case, the 'break-even' costs (where both offers are equally attractive) are
simply equal to the costs of the current best offer

  [ ifelse breed = Consuming-Businesses
    [ let break-even-profit-per-btu 0

      if (effective-relative-wtp-for-loyalty + effective-relative-wtp-for-fit-difference != -1)  ;; prevent divide by 0 error
      [ set break-even-profit-per-btu (a_best-offer-profit-per-btu + a_best-offer-virtual-loyalty-bonus) / ( effective-relative-wtp-for-loyalty +
effective-relative-wtp-for-fit-difference + 1 ) ]
          ;; this is the formula to solve the equation where both offers are equal after balancing profit, loyalty and preference fit. Elaboration can
be found in the Excel file 'market research formula....xlx'

          ;; since wtp for preference fit is relative to the HIGHEST profit of the two offers, this first formula assumes that the profit of the new
offer is higher than that of the current best offer.
          ;; if this turns out not to be the case, use a second formula to determine the actual switching profit value:
      if (effective-relative-wtp-for-loyalty + effective-relative-wtp-for-fit-difference = -1) or (break-even-profit-per-btu < a_best-offer-profit-per-
btu)
      [ set break-even-profit-per-btu (a_best-offer-profit-per-btu + a_best-offer-virtual-loyalty-bonus - a_best-offer-profit-per-btu * effective-
relative-wtp-for-fit-difference) / (effective-relative-wtp-for-loyalty + 1) ]

      ;show (word "tick " current-tick ": CB market research. a_best-offer-profit-per-btu: " a_best-offer-profit-per-btu "  a_best-offer-virtual-
loyalty-bonus: " a_best-offer-virtual-loyalty-bonus "  effective wtp for loyalty:  " effective-relative-wtp-for-loyalty "  effective-relative-wtp-for-
fit-difference: " effective-relative-wtp-for-fit-difference "  break-even-profit-per-btu: " break-even-profit-per-btu)

      ;; the agent will never accept a negative profit
      if break-even-profit-per-btu < 0 [ set break-even-profit-per-btu 0 ]

      ;; if the new offer has a lower max preference threshold violation, the agent is willing to pay any price where she still makes any profit
      if max-Preference-threshold-violation < a_best-offer-preference-violation [ set break-even-profit-per-btu 0 ]

      ;; calculate implied maximum cost for the calculated minimum profit for a switch
      let total-revenue-per-btu a_default-revenue-per-btu * [a_revenue-multiplier] of new-CM-lib
      set break-even-costs-per-btu total-revenue-per-btu - a_default-cost-per-btu - break-even-profit-per-btu

      ;if g_display-CB-actions? [ show (word "tick " current-tick ": CB market research, break-even-profit: " break-even-profit-per-btu "  best offer
profit: " a_best-offer-profit-per-btu) ]
    ]
    [ if (effective-relative-wtp-for-loyalty + effective-relative-wtp-for-fit-difference != 1)  ;; prevent divide by 0 error
      [ set break-even-costs-per-btu (a_best-offer-costs-per-btu - a_best-offer-virtual-loyalty-bonus) / ( 1 - effective-relative-wtp-for-loyalty -
effective-relative-wtp-for-fit-difference ) ]
          ;; same as above, the formula is elaborated in the specified excel file.

          ;; the formula assumes here that the cost of the new offer is lower than that of the old offer. If not, use a second formula:
      if (effective-relative-wtp-for-loyalty + effective-relative-wtp-for-fit-difference = 1) or (break-even-costs-per-btu > a_best-offer-costs-per-
btu)
      [ set break-even-costs-per-btu (a_best-offer-costs-per-btu - a_best-offer-virtual-loyalty-bonus + a_best-offer-costs-per-btu * effective-
relative-wtp-for-fit-difference) / (1 - effective-relative-wtp-for-loyalty) ]

      ;; if the new offer has a lower max preference threshold violation, the agent is willing to pay any price within her budget
      if max-Preference-threshold-violation < a_best-offer-preference-violation [ set break-even-costs-per-btu a_max-cost-threshold-for-unit-of-Need *
a_Need-per-btu ]

      ;; now it could be the case that the break-even-costs are above the cost threshold. In that case, set the break-even-cost equal to the cost
threshold
      if break-even-costs-per-btu / a_Need-per-btu > a_max-cost-threshold-for-unit-of-Need
      [ set break-even-costs-per-btu a_max-cost-threshold-for-unit-of-Need * a_Need-per-btu ]
    ]
  ]
  report break-even-costs-per-btu
end

to select-best-CM-from-set [ CM-set forbidden-SM-1 forbidden-SM-2 for-market-research? ]  ;; CB/C procedure. 'CM-set' is a turtle-set of allowed CM's
to choose from, based on offers on the market and infrastructure availability of the agent
  ;; The CMs are considered according to structured hierarchical rules: first, always choose the one with the lowest budget violation
  ;; If there are multiple options within the budget, always choose the one with the lowest Preference threshold violation
  ;; If there are multiple options within all Preference thresholds, best combination of cost (or profit) and lifestyle/strategic fit, based on the
willingness to pay

  let allowed-CMs-list sort-on [random 10] CM-set  ;; this is just to convert the CM-set from a turtle-set to a list
  let CM-with-lowest-budget-violation nobody
  let current-cost-per-unit-of-Need-of-CM-with-lowest-budget-violation 100000000

  ; use local variables to keep track of (properties of) the best offer/CM encountered so far, while going through all allowed CMs. The values will be
updated automatically by the first offer encountered, and then iteratively compared to other offers.
  let overall-best-offer-SM -1
  let overall-best-offer-cost-per-btu -1
  let overall-best-offer-virtual-loyalty-bonus -1
  let overall-best-offer-profit-per-btu -1
  let overall-best-offer-preference-violation -1
  let overall-best-offer-fit -1

  let old-CM a_Consumption-Model

  ;; for each CM-lib under consideration, calculate the switching costs and the lifestyle/strategic fit score, then compare it with the best CM found
so far
```

```
  foreach allowed-CMs-list
  [ ;; within this loop, we compare several CMs. We find the best input offer for each CM and use the 'a_best-offer-xxx' attributes of CB/C to
temporarily store the properties of that offer.
    ;; the 'overall-best-offer-xxx' local variables then store the properties of the overall best offer encountered so far. That offer is then used for
new comparisons to find the best offer.
    let possible-new-CM ?

    ;; Find best offer on the market for this CM. This updates a_best-offer-xxx attributes of the CB/C, which can then be used by 'compare-best-offer-
generic'
    find-best-input-offer-for-given-CM possible-new-CM for-market-research? false forbidden-SM-1 forbidden-SM-2  ;;'false' indicates that the full
strategic reconsideration cycle should be taken into account here

    ;; compare the best offer for the CM we're now looking at with the best offer we encountered so far
    if is-agent? overall-best-offer-SM
    [ compare-with-best-offer-generic overall-best-offer-SM overall-best-offer-cost-per-btu overall-best-offer-profit-per-btu overall-best-offer-
preference-violation overall-best-offer-fit overall-best-offer-virtual-loyalty-bonus  ]

    ;; if we found a new overall best offer, store its properties for comparison with all following offers.
    if a_best-offer-SM != overall-best-offer-SM
    [ set overall-best-offer-SM a_best-offer-SM
      set overall-best-offer-cost-per-btu a_best-offer-costs-per-btu
      set overall-best-offer-virtual-loyalty-bonus a_best-offer-virtual-loyalty-bonus
      if (breed = Consuming-Businesses) [ set overall-best-offer-profit-per-btu a_best-offer-profit-per-btu ]
      set overall-best-offer-preference-violation a_best-offer-preference-violation
      set overall-best-offer-fit a_best-offer-fit
    ]
  ]

  if (overall-best-offer-SM = nobody) [ show allowed-CMs-list error "best offer is nobody!" ]

  ;; After comparing all offers, store the properties of the overall best offer back into the 'a_best-offer-xxx' properties of the CB/C
  update-best-offer-attributes overall-best-offer-SM overall-best-offer-cost-per-btu overall-best-offer-virtual-loyalty-bonus overall-best-offer-
profit-per-btu overall-best-offer-fit overall-best-offer-preference-violation

  if (g_display-CB-actions? and (breed = Consuming-Businesses))
  [ show (word "tick " current-tick ": Select best CM from set. ID of best: " [a_id-within-type] of one-of consumption-models-lib with [a_main-input-id
= [a_main-output-id] of [a_best-offer-SM] of myself ] "  cost per btu: " a_best-offer-costs-per-btu "  loyalty bonus: " a_best-offer-virtual-loyalty-
bonus "  fit: " a_best-offer-fit) ]
end

to compare-with-best-offer-generic [ new-offer new-offer-total-cost-per-btu new-offer-profit-per-btu new-offer-max-Preference-threshold-violation new-
offer-preference-fit new-offer-virtual-loyalty-bonus ] ; CB/C procedure
  ;; generic procedure to let a CB/C compare properties of a new offer with the offer currently identified as best

  let possible-new-CM-lib one-of Consumption-Models-lib with [ a_main-input-id = [a_main-output-id] of new-offer ]
  let new-offer-cost-per-unit-of-Need new-offer-total-cost-per-btu / a_Need-per-btu
  let best-offer-cost-per-unit-of-Need a_best-offer-costs-per-btu / a_Need-per-btu

  let any-budget-violation? false
  let new-offer-is-best? false
  if not is-agent? a_best-offer-SM [ set new-offer-is-best? true ]

  if not new-offer-is-best?
  [ ;; FIRST COMPARISON RULE
    ;; The budget is a hard constraint: if both offers violate the budget, the offer with the lowest budget violation is always chosen as best offer.
    ;; If both offers are within the budget,  the procedure will continue to a further comparison that also takes into account the agent's Preferences
    ifelse breed = Consumers
    [ if (new-offer-cost-per-unit-of-Need > a_max-cost-threshold-for-unit-of-Need) or (best-offer-cost-per-unit-of-Need > a_max-cost-threshold-for-
unit-of-Need)
      [ set any-budget-violation? true
        set new-offer-is-best? (new-offer-cost-per-unit-of-Need < best-offer-cost-per-unit-of-Need)
      ]
    ]
    ;; for Consuming-Businesses, the 'budget' violation concerns whether the CB can still make any profit
    [ if (new-offer-profit-per-btu < 0) or (a_best-offer-profit-per-btu < 0)
      [ set any-budget-violation? true
        set new-offer-is-best? new-offer-profit-per-btu > a_best-offer-profit-per-btu
      ]
    ]

    ;; SECOND COMPARISON RULE: preference threshold violation
    if not any-budget-violation?
    [ ;; evaluate CM threshold violations of Preferences
      let any-preference-threshold-violation? (new-offer-max-Preference-threshold-violation > 0) or (a_best-offer-preference-violation > 0)
      ifelse any-preference-threshold-violation?
      [ set new-offer-is-best? new-offer-max-Preference-threshold-violation < a_best-offer-preference-violation ]

      ;; THIRD COMPARISON RULE
      [ ;; if this is not the first offer found that respects all thresholds, compare it with the best offer found so far by weighing money aspects
against non-money aspects (i.e. lifestyle/strategic fit and loyalty score)
        let virtual-loyalty-bonus-difference new-offer-virtual-loyalty-bonus - a_best-offer-virtual-loyalty-bonus
        let fit-difference new-offer-preference-fit - a_best-offer-fit

        ;; if the agent is a Consuming Business:
        ifelse breed = Consuming-Businesses
        [ ;; compare the values for the 'current best offer' (for the CM now under consideration) with the values for the overall best offer found so
far
          let profit-difference-per-btu new-offer-profit-per-btu - a_best-offer-profit-per-btu
          let highest-possible-profit max (list new-offer-profit-per-btu a_best-offer-profit-per-btu)

          ;; calculate the monetary equivalent of the better (or worse) strategic fit, relative to the highest possible profit, and compare it with the
difference in profit
          if profit-difference-per-btu + virtual-loyalty-bonus-difference + fit-difference * a_relative-wtp-for-preference-fit * highest-possible-
profit > 0
          [ set new-offer-is-best? true ]
        ]

        ;; if the agent is a Consumer:
```

```
        [ ;; compare the values for the 'current best offer' (for the CM now under consideration) with the values for the overall best offer found so
far
          let cost-difference-per-btu new-offer-total-cost-per-btu - a_best-offer-costs-per-btu
          let lowest-possible-cost min (list new-offer-total-cost-per-btu a_best-offer-costs-per-btu)

          ;; calculate the monetary equivalent of the better (or worse) lifestyle fit, relative to the lowest possible cost, and compare it with the
difference in profit
          if fit-difference * a_relative-wtp-for-preference-fit * lowest-possible-cost - cost-difference-per-btu + virtual-loyalty-bonus-difference > 0
          [ set new-offer-is-best? true ]
        ]
      ]
    ]
  ]

  if new-offer-is-best?
  [ update-best-offer-attributes new-offer new-offer-total-cost-per-btu new-offer-virtual-loyalty-bonus new-offer-profit-per-btu new-offer-preference-
fit new-offer-max-Preference-threshold-violation ]

  if (g_display-CB-actions? and (breed = Consuming-Businesses))
  [ show (word "tick " current-tick ": Compare new best offer on market. ID of best: " [a_id-within-type] of one-of consumption-models-lib with
[a_main-input-id = [a_main-output-id] of [a_best-offer-SM] of myself ] "  cost per btu: " a_best-offer-costs-per-btu "  loyalty bonus: " a_best-offer-
virtual-loyalty-bonus "  fit: " a_best-offer-fit) ]
end

to update-best-offer-attributes [ new-offer new-offer-total-cost-per-btu new-offer-virtual-loyalty-bonus new-offer-profit-per-btu new-offer-fit new-
offer-preference-violation ]  ;; CB/C procedure
  ;; small procedure to update the CB/C attributes relating to her currently identified best offer

  set a_best-offer-SM new-offer
  set a_best-offer-costs-per-btu new-offer-total-cost-per-btu
  set a_best-offer-virtual-loyalty-bonus new-offer-virtual-loyalty-bonus
  if (breed = Consuming-Businesses) [ set a_best-offer-profit-per-btu new-offer-profit-per-btu  ]
  set a_best-offer-fit new-offer-fit
  set a_best-offer-preference-violation new-offer-preference-violation
end

to-report filter-on-infrastructure-availability [ model-set ]  ;; PB/CB/C procedure
  ;; filters the specified set of MMs/SMs/CMs on those that the agent has the required infrastructures available for

  let root-calling-agent self

  ;; traverse all models in the 'model-set'
  ask model-set
  [ let all-infrastructures-available? true

    ;; traverse over the required infrastructures for this model and check if the agent has availability of this infrastructure
    foreach a_required-infrastructure-ids-list
    [ if not (member? ? ([a_available-infrastructure-ids-list] of root-calling-agent))  ;; 'a_available-infrastructure-ids-list' is a list of the
available infrastructures of the agent.
                                                                                        ;; '?' is the required infrastructure we're currently checking
availability for
      [ set all-infrastructures-available? false ]
    ]

    ;; if the agent does not have availability of all required infrastructures, remove the model from the set
    if not all-infrastructures-available? [ set model-set remove-from-turtle-set self model-set ]
  ]
  report model-set
end

to-report calculate-total-net-switching-costs [ old-model new-model-lib for-market-research? ]  ;; PB/CB/C procedure
  ;; this procedure calculates the effective net one-time switching costs for switching from 'old-model' to 'new-model-lib', from the perspective of a
specific agent.
  ;; the old-model is an object instance, the new-model-lib is a library object

  let old-model-id ifelse-value is-agent? old-model [ [a_library-id] of old-model ] [ 0 ]
  let new-model-id [a_library-id] of new-model-lib
  let total-switching-costs 0

  ;; switching costs are only relevant if the new model is different than the old one
  if old-model-id != new-model-id
  [ ;; one part of the switching costs is the actual one-time investment cost specified for the new model (only relevant if it is different than the
agent's current model)
    let one-time-investment-cost [a_initial-investment-cost] of new-model-lib

    ;; when switching, the old stocks can be sold to the world market or have to be disposed (possibly against a cost). Calculate the net revenue
(costs are a negative value here)
    let revenue-from-selling-obsolete-stocks 0

    ifelse (breed = Producing-Businesses)
    [ if is-agent? old-model
      [ set revenue-from-selling-obsolete-stocks calculate-revenue-from-selling-obsolete-stocks ifelse-value ([breed] of old-model = Sales-Models) [
[a_main-input-stock] of old-model ] [ (turtle-set) ] ;; now we can finally calculate the expected revenue when selling the remaining stock after time-
until-switch
        set revenue-from-selling-obsolete-stocks calculate-revenue-from-selling-obsolete-stocks [a_main-output-stock] of old-model
      ]
    ]
    [ if (not for-market-research?) and is-agent? old-model
      [ set revenue-from-selling-obsolete-stocks revenue-from-selling-obsolete-stocks + calculate-revenue-from-selling-obsolete-stocks [a_associated-
owned-products] of old-model ]
    ]

    ;; if the new model requires connection to certain infrastructures that the agent is not currently connected to, calculate the total costs of
making those new connections
    let infrastructure-connection-cost 0
    foreach [a_required-infrastructure-ids-list] of new-model-lib
    [ let infrastructure-id ?
      if not member? infrastructure-id a_connected-infrastructure-IDs-list
      [ set infrastructure-connection-cost [a_one-time-connection-fee] of infrastructure-provider-with-id infrastructure-id ]
```

```
    ]

    ;; if the new model requires the possession of skills that the agent does not currently has, calculate the total costs of acquiring those new
skills
    let skill-acquisition-cost 0
    foreach [a_required-skill-ids-list] of new-model-lib
    [ let skill-id ?
      if not member? skill-id a_acquired-skill-IDs-list
      [ set skill-acquisition-cost [a_learning-costs] of skill-object-with-id skill-id ]
    ]

    set total-switching-costs one-time-investment-cost - revenue-from-selling-obsolete-stocks + infrastructure-connection-cost + skill-acquisition-cost
  ]

  report total-switching-costs
end

to switch-to-best-offer   ;; CB/C procedure
  ;; procedure to switch to the offer (and associated CM) that has been identified as best option by previous procedures

  let main-input-id [a_main-output-id] of a_best-offer-SM
  let CM-id [a_library-id] of one-of Consumption-Models-lib with [ a_main-input-id = main-input-id ]
  let CM-change? (not is-agent? a_Consumption-Model) or (([a_library-id] of a_Consumption-Model) != CM-id)

  ;; notify the supplier that this agent is no longer a customer
  if (is-agent? a_Consumption-Model) [ disconnect-from-current-supplier ]

  if CM-change?
  [ remove-current-CM
    set a_cash-balance a_cash-balance - [a_initial-investment-cost] of library-object-with-id CM-id
    adopt-new-CM CM-id
    acquire-needed-skills-and-infrastructures-for-new-model CM-id
    if current-tick > 0 [ update-needed-n-new-products ]
  ]

  ;; notify the supplier that there is a new customer
  connect-to-new-supplier a_best-offer-SM
  ask a_consumption-model [ set color [color] of [a_current-suppliers-SM] of a_owner ]

  set a_current-offer-total-costs-per-btu a_best-offer-costs-per-btu
  set a_current-offer-fit a_best-offer-fit
end

to remove-current-CM  ;; CB/C procedure
  ;; if a consuming agent changes to a new CM, she first removes her current CM, meaning that it disappears from the model and remaining stocks are
disposed of (either to WM or PE)

  let root-agent self
  if is-agent? a_Consumption-Model
  [ ;; dispose of remaining stocks and update the cash balance accordingly
    ask a_Consumption-Model
    [ ask a_associated-owned-products
      [ let disposal-cost calculate-cost-of-product-disposal a_library-id remaining-amount
        ask root-agent [ set a_cash-balance a_cash-balance - disposal-cost ]
        die  ;; the products disappear from the model
      ]
      die  ;; the CM disappears from the model
    ]
  ]
end

to adopt-new-CM [ CM-id ]  ;; CB/C procedure
  ;; this procedure creates a new CM of the specified type and connects it to the calling agent. It also initialises some attributes and notifies the
supplier (specified by 'a_best-offer-SM')

  let owner self

  ;; create a new CM and initialise attributes
  ask library-object-with-id CM-id
  [ hatch-Consumption-Models 1  ;; the library object 'hatches' (creates) a new CM, copying all same-called attributes. The created CM then executes
the following actions:
    [ set a_owner owner
      visualise-model
      ask owner [ set a_Consumption-Model myself ]   ;; make the connection here between the root calling consuming agent and the newly created
Consumption Model
      set a_library-object myself   ;; 'myself' here refers to the consumption-model-lib(rary) object that was asked to hatch a new consumption-model
to be adopted by the root calling CB or C
      set a_associated-owned-products (turtle-set)

      if a_main-input-is-Tool? [ set a_associated-owned-products (turtle-set) ]   ;; make sure here that the variable a_associated-owned-products is
seen by NetLogo as a (so far empty) turtle-set from now on.
                                                                                  ;; This is necessary to later add turtles to that turtle-set
      if a_main-input-is-Consumable? [ set a_associated-owned-products (turtle-set (make-initial-empty-Consumable true)) ]

      set a_consumed-main-input-volume-per-btu a_main-input-ratio-in-full-unit-equivalents * ([a_Need-per-btu] of a_owner)  ;; calculate the consumed
main input volume per btu based on the Need of the owner
    ]
  ]

  set-color-CA
  if breed = Consuming-Businesses   ;; if the calling agent is a CB, she should also update her revenue per btu based on the properties of the newly
adopted CM
  [ set a_actual-revenue-per-btu a_default-revenue-per-btu * [a_revenue-multiplier] of a_Consumption-Model ]

  ;; notification of new model adoption
  ;if current-tick > 0 [ show (word "tick " ticks ": NEW MODEL ADOPTED: " [a_id-within-type] of a_Consumption-Model) ]
end

to acquire-needed-skills-and-infrastructures-for-new-model [ model-id ]  ;; PB/CB/C procedure
```

```
  ;; this procedure makes sure that the agent acquires the additional infrastructures and skills needed to use the new model (specified as an argument)
  ;; model-id can relate to either a MM, SM or CM

  let model-library-object library-object-with-id model-id

  ;; acquire additional Infrastructures
  foreach [a_required-infrastructure-ids-list] of model-library-object
  [ let infrastructure-id ?
    if not member? infrastructure-id a_connected-infrastructure-IDs-list
    [ let infrastructure-connection-cost [a_one-time-connection-fee] of infrastructure-provider-with-id infrastructure-id
      set a_cash-balance a_cash-balance - infrastructure-connection-cost
      set a_connected-infrastructure-IDs-list lput infrastructure-id a_connected-infrastructure-IDs-list
    ]
  ]

  ;; acquire additional Skills
  foreach [a_required-skill-ids-list] of model-library-object
  [ let skill-id ?
    if not member? skill-id a_acquired-skill-IDs-list
    [ let skill-acquisition-cost [a_learning-costs] of skill-object-with-id skill-id
      set a_cash-balance a_cash-balance - skill-acquisition-cost
      set a_acquired-skill-IDs-list lput skill-id a_acquired-skill-IDs-list
    ]
  ]
end

to adopt-capacity [ new-capacity ]  ;; PB procedure
  ;; this procedure makes sure that the PB adopts a certain new capacity level and updates the structural costs to be paid

  ;; adjust the capacity of the Manufacturing Model to the specified new level
  ask a_MM
  [ set a_output-capacity-per-btu new-capacity
    set a_structural-costs-per-btu new-capacity * [ a_structural-costs-per-unit-of-capacity ] of a_library-object
  ]
end

to set-SM-price [ SM-to-update new-prices ]  ;; PB procedure
  ;; adjust the prices of one of the (maximum 2) SMs of the PB according to the 'new-prices' (a list with three values: one-time contract cost, cost
per unit, cost per btu)

  ask SM-to-update
  [ set a_offer-price-one-time-contract-cost item 0 new-prices
    set a_offer-price-per-unit item 1 new-prices
    set a_offer-price-per-btu item 2 new-prices
    if a_main-output-is-Service?
    [ ask Service-Contracts
      [ if a_seller = [a_owner] of myself
        [ set a_price-per-unit  [a_offer-price-per-unit] of myself ]
      ]
    ]
  ]
end

to disconnect-from-current-supplier  ;; CB/C procedure
  ;; this procedure does two things: it removes the associated Service Contract (if applicable) and it makes sure the supplier updates her production
volumes

  if is-agent? a_current-suppliers-SM and [not a_main-input-is-Tool?] of a_Consumption-Model
  [ let daily-volume a_Need-per-btu * [ a_main-input-ratio ] of a_Consumption-Model
    ask a_current-suppliers-SM
    [ set a_contracted-output-volume-per-btu a_contracted-output-volume-per-btu - daily-volume
      let converted-volume daily-volume * a_main-input-ratio-in-full-unit-equivalents
      ask a_associated-MM [ set a_contracted-output-volume-per-btu a_contracted-output-volume-per-btu - converted-volume ]
      update-target-stock-size a_contracted-output-volume-per-btu
    ]
  ]

  if [a_main-input-is-Service?] of a_Consumption-Model  [ ask a_service-contract [ die ] ]

  set a_current-supplier nobody
  set a_current-suppliers-SM nobody
end

to connect-to-new-supplier [ new-suppliers-SM ]  ;; CB/C procedure
  ;; this procedure does three things: it updates the 'a-current-suppliers-SM' attribute, creates the associated Service Contract (if applicable)
  ;; and it makes sure the supplier updates her production volumes

  set a_current-supplier [a_owner] of new-suppliers-SM
  set a_current-suppliers-SM new-suppliers-SM

  if [a_main-output-is-Service?] of a_best-offer-SM [ sign-service-contract new-suppliers-SM ]

  if [not a_main-input-is-Tool?] of a_Consumption-Model
  [ let daily-volume a_Need-per-btu * [ a_main-input-ratio ] of a_Consumption-Model
    ask a_current-suppliers-SM
    [ set a_contracted-output-volume-per-btu a_contracted-output-volume-per-btu + daily-volume
      let converted-volume daily-volume * a_main-input-ratio-in-full-unit-equivalents
      ask a_associated-MM [ set a_contracted-output-volume-per-btu a_contracted-output-volume-per-btu + converted-volume ]
      update-target-stock-size a_contracted-output-volume-per-btu
    ]
  ]
end

to sign-service-contract [ associated-SM ]  ;; CB/C procedure
  ;; this procedure creates a Service Contract between the calling agent and the owner of the specified 'associated-SM'

  let daily-volume a_Need-per-btu * [ a_main-input-ratio ] of a_Consumption-Model

  hatch-Service-Contracts 1
```

```
  [ set a_seller [ a_owner ] of associated-SM
    set a_buyer myself
    set a_contracted-object [ a_main-output-ID ] of associated-SM

    set a_price-per-btu [ a_offer-price-per-btu ] of associated-SM
    set a_price-per-unit [ a_offer-price-per-unit ] of associated-SM

    ask a_buyer
    [ set a_service-contract myself
      pay-seller-one-time-fee
    ]

    ;; visualisation stuff
    move-to a_buyer   set size 0.6   show-turtle   set heading 90   fd 0.5   set heading 0   fd 0.5
    set color [color] of library-object-with-id a_contracted-object
  ]
end

to reconsider-price-and-target-stock-size [ SM-P SM-S capacity-fixed? hypothetical-offers? CA-sample ] ;; PB procedure
  ;; this procedure lets a PB optimise the prices and target stock sizes of her SM(s), eventually based on market research.
  ;; if both SMs are present, the optimisation involves an iterative procedure

  set a_time-of-next-tactical-reconsideration current-tick + a_tactical-reconsideration-interval
  let asking-PB self
  set opt_CA-sample CA-sample

  set opt_best-net-profit-P-for-this-config 0
  set opt_best-net-profit-S-for-this-config 0

  set opt_best-volume-P-for-this-config ifelse-value (is-agent? SM-P) [ [ a_main-input-ratio-in-full-unit-equivalents * sum [ [a_consumed-main-input-
volume-per-btu] of a_Consumption-Model ] of g_all-consuming-agents with [ a_current-suppliers-SM = myself ] ] of SM-P ] [ 0 ]
  set opt_best-volume-S-for-this-config ifelse-value (is-agent? SM-S) [ [ a_main-input-ratio-in-full-unit-equivalents * sum [ [a_consumed-main-input-
volume-per-btu] of a_Consumption-Model ] of g_all-consuming-agents with [ a_current-suppliers-SM = myself ] ] of SM-S ] [ 0 ]

  let simultaneous-optimisation? (is-agent? SM-P and is-agent? SM-S)

  if g_display-market-research-numbers? or (g_display-simultaneous-optimisation-updates? and simultaneous-optimisation?)  [show "" show "new price
reconsideration!" show ""]

  ;; consuming agents have to compare the offers of the asking PB with the best of all OTHER offers on the market in order to provide a switch price.
  ;; So, if their current best offer is offered by the asking PB, they first have to find out what their fallback option is (from another supplier!) so
they can evaluate at what price they would switch.
  prepare-CA-sample-for-market-research asking-PB SM-P SM-S

  ifelse simultaneous-optimisation?
  [ do-iterative-price-optimisation SM-P SM-S a_ROI-period capacity-fixed? ]
  [ let SM-to-update ifelse-value is-agent? SM-P [SM-P] [SM-S]

    ;; one-time price optimisation
    optimise-price-for-one-SM SM-to-update nobody a_ROI-period capacity-fixed?
    if g_display-simultaneous-optimisation-updates?
    [ show (word "new price P after single optimisation: " (ifelse-value is-agent? SM-P [ [a_offer-price-per-unit] of SM-P ] [ "N/A" ]) "  new price S:
" (ifelse-value is-agent? SM-S [ [a_offer-price-per-unit] of SM-S ] [ "N/A" ]) "  volumes P and S: " opt_best-volume-P-for-this-config " " opt_best-
volume-S-for-this-config "  net profit P: " opt_best-net-profit-P-for-this-config "  net profit S: " opt_best-net-profit-S-for-this-config "  total: "
opt_best-profit-for-this-config) ]

  ]

  ask CA-sample
  [ reset-best-offer-properties-from-backup
    set a_intermediate-best-offer-SM-during-MR 0
  ]

  if not hypothetical-offers?
  [ ask g_all-consuming-agents
    [ if (not is-agent? a_best-offer-SM) or (asking-PB = ([a_owner] of a_best-offer-SM))
      [ reconsider-consumption-strategy true SM-P SM-S ]

      if is-agent? SM-P [ compare-new-offer-on-market-with-best-offer SM-P ]
      if is-agent? SM-S [ compare-new-offer-on-market-with-best-offer SM-S ]
    ]
  ]
end

to prepare-CA-sample-for-market-research [ asking-PB SM-P SM-S ]  ;; PB procedure
  ;; procedure to make sure all CB/Cs in the CA-sample 'backup' the properties of their currently identified best offer, and reset some other
attributes

  ask opt_CA-sample
  [ if (is-agent? a_best-offer-SM) [ update-best-offer-preference-violation-and-fit a_best-offer-SM ]
    if (not is-agent? a_best-offer-SM) or (asking-PB = ([a_owner] of a_best-offer-SM))
    [ reconsider-consumption-strategy true SM-P SM-S ]
    backup-best-offer-properties
    set a_intermediate-best-offer-SM-during-MR 0
    set a_last-market-research-volume 0
  ]
end

to do-iterative-price-optimisation [ SM-P SM-S horizon capacity-fixed? ] ; PB procedure
  ;; procedure to iteratively optimise both SMs of the PB, taking into account substitution effects between them

  let first-iteration? true
  let best-overall-net-profit 0.00000000000001
  let new-net-profit 0.00000000000001
  set opt_best-profit-for-this-config best-overall-net-profit

  let iteration-nr 0

  ;; iterative price optimisation
```

```
  while [ first-iteration? or (((new-net-profit - best-overall-net-profit) / best-overall-net-profit) >= 0.025) ]
  [ set best-overall-net-profit new-net-profit

    ask opt_CA-sample
    [ set a_intermediate-best-offer-SM-during-MR a_best-offer-SM

      let latest-best-offer a_best-offer-SM
      if a_best-offer-SM = SM-P [ reset-best-offer-properties-from-backup ]
      if display-additional-sim-opt-updates? [ show (word "latest-best-offer: " latest-best-offer "  best offer after possible backup: " a_best-offer-
SM "  SM-P: " SM-P "  SM-S: " SM-S) ]
      if (not (a_best-offer-SM = SM-S)) [ compare-new-offer-on-market-with-best-offer SM-S ]
    ]

    let other-SM ifelse-value first-iteration? [ nobody ] [ SM-S ]  ;; in the first iteration, we want to optimise the volume of the product-based SM
independently of the service-based SM.
                                                                    ;; To do this, we run the 'find-best-price..' procedure with 'nobody' as second
input argument, meaning that the procedure ignores the other SM of this PB
    optimise-price-for-one-SM SM-P other-SM horizon capacity-fixed?

    ask opt_CA-sample [ compare-new-offer-on-market-with-best-offer SM-P ]

    if g_display-simultaneous-optimisation-updates?
    [ show (word "new price P after step 1: " [a_offer-price-per-unit] of SM-P "  new price S: " [a_offer-price-per-unit] of SM-S "  volumes P and S: "
opt_best-volume-P-for-this-config " " opt_best-volume-S-for-this-config "  net profit P: " opt_best-net-profit-P-for-this-config "  net profit S: "
opt_best-net-profit-S-for-this-config "  total: " opt_best-profit-for-this-config) ]

    ask opt_CA-sample
    [ if not first-iteration? [ set a_intermediate-best-offer-SM-during-MR a_best-offer-SM ]

      let latest-best-offer a_best-offer-SM
      if (a_best-offer-SM = SM-S) [ reset-best-offer-properties-from-backup ]
      if display-additional-sim-opt-updates? [ show (word "latest-best-offer: " latest-best-offer "  best offer after possible backup: " a_best-offer-
SM "  SM-P: " SM-P "  SM-S: " SM-S) ]
      if (not (a_best-offer-SM = SM-P)) [ compare-new-offer-on-market-with-best-offer SM-P ]
    ]

    optimise-price-for-one-SM SM-S SM-P horizon capacity-fixed?

    ask opt_CA-sample [ compare-new-offer-on-market-with-best-offer SM-S ]

    if g_display-simultaneous-optimisation-updates?
    [ show (word "new price P after step 2: " [a_offer-price-per-unit] of SM-P "  new price S: " [a_offer-price-per-unit] of SM-S "  volumes P and S: "
opt_best-volume-P-for-this-config " " opt_best-volume-S-for-this-config "  net profit P: " opt_best-net-profit-P-for-this-config "  net profit S: "
opt_best-net-profit-S-for-this-config "  total: " opt_best-profit-for-this-config) ]

    set new-net-profit opt_best-profit-for-this-config

    set first-iteration? false

    set iteration-nr iteration-nr + 1
  ]
end



to initialise-opt-variables-for-new-optimisation [ SM-to-update other-SM-of-PB horizon capacity-fixed?] ;; PB procedure
  ;; This procedure prepares for a price optimisation by calculating or resetting all kinds of relevant variables

  set opt_sorted-filtered-MR-results-nlist 0
  set opt_SM-to-update SM-to-update
  set opt_other-SM-of-PB other-SM-of-PB
  set opt_associated-MM [a_associated-MM] of opt_SM-to-update
  set opt_horizon horizon
  set opt_capacity-fixed? capacity-fixed?

  set opt_consider-other-SM? is-agent? opt_other-SM-of-PB

  set opt_this-SM-input-to-output-ratio [a_main-input-ratio-in-full-unit-equivalents] of SM-to-update
  set opt_current-price [a_offer-price-per-unit] of opt_SM-to-update
  set opt_offer-price-list [concatenated-offer-price-list] of opt_SM-to-update  ;; we only need the first and last value of this list in subprocedures,
but storing the whole list is easier
  set opt_variable-cost-per-unit calculate-variable-cost-per-unit-SM opt_SM-to-update
  set opt_price ifelse-value (opt_current-price < opt_variable-cost-per-unit) [ opt_variable-cost-per-unit ] [ opt_current-price ]
  set opt_min-price 0 ; current-price * 0.7
  set opt_max-price 100000000000000 ; min-price / 0.7 * 1.1   ; this suggestion will currently give trouble during PB strategic reconsideration:
current-price of a ghost SM is always 0.
  if opt_min-price < opt_variable-cost-per-unit [ set opt_min-price min (list opt_max-price opt_variable-cost-per-unit) ]
  set opt_effective-input-volume 0
  set opt_new-total-capacity [a_output-capacity-per-btu] of opt_associated-MM

  set opt_risk-factor-this-SM ifelse-value ((SM-to-update = a_product-based-SM) or (SM-to-update = a_service-based-SM))  [0] [a_market-research-risk-
factor]
  set opt_risk-factor-other-SM ifelse-value ((other-SM-of-PB = a_product-based-SM) or (SM-to-update = a_service-based-SM))  [0] [a_market-research-
risk-factor]
  set opt_cumulative-input-volume-for-this-SM 0

  ;; calculate the free MM capacity, based on the required production volume to accommodate the expected sales for the other SM
  let capacity horizon * [a_output-capacity-per-btu] of opt_associated-MM
  set opt_free-capacity max (list 0 (capacity - opt_cumulative-input-volume-for-other-SM))

  ifelse opt_consider-other-SM?
  [ set opt_cumulative-input-volume-for-other-SM ifelse-value ([a_main-output-is-Service?] of other-SM-of-PB) [ opt_best-volume-S-for-this-config ] [
opt_best-volume-P-for-this-config ]
    set opt_other-SM-input-to-output-ratio [a_main-input-ratio-in-full-unit-equivalents] of other-SM-of-PB

    ;; calculate the gross profit for the other-SM-of-PB: this is necessary to calculate lost profits later on in the procedure.
    let variable-cost-per-unit-of-other-SM calculate-variable-cost-per-unit-SM other-SM-of-PB
    let gross-profit-per-sold-output-of-other-SM [ a_offer-price-per-unit ] of other-SM-of-PB - variable-cost-per-unit-of-other-SM
    set opt_gross-profit-per-MM-output-unit-of-other-SM gross-profit-per-sold-output-of-other-SM / opt_other-SM-input-to-output-ratio
```

```
  ]
  [ set opt_cumulative-input-volume-for-other-SM 0
    set opt_other-SM-input-to-output-ratio 0
    set opt_gross-profit-per-MM-output-unit-of-other-SM 0
  ]

  set opt_net-profit-of-other-SM ifelse-value ([a_main-output-is-Service?] of SM-to-update) [opt_best-net-profit-P-for-this-config] [opt_best-net-
profit-S-for-this-config]

  let best-profit-P-after-last-round opt_best-net-profit-P-for-this-config
  let best-profit-S-after-last-round opt_best-net-profit-S-for-this-config

  set opt_best-price-P-for-this-config ifelse-value ((is-agent? a_product-based-SM) and ((a_product-based-SM = SM-to-update) or (a_product-based-SM =
other-SM-of-PB))) [ [ a_offer-price-per-unit ] of a_product-based-SM ] [ 0 ]
  set opt_best-price-S-for-this-config ifelse-value ((is-agent? a_service-based-SM) and ((a_service-based-SM = SM-to-update) or (a_service-based-SM =
other-SM-of-PB))) [ [ a_offer-price-per-unit ] of a_service-based-SM ] [ 0 ]
  set opt_best-profit-for-this-config 0
  set opt_best-capacity-for-this-config [a_output-capacity-per-btu] of opt_associated-MM
  set opt_best-net-profit-P-for-this-config 0
  set opt_best-net-profit-S-for-this-config 0
  set opt_best-volume-P-for-this-config 0
  set opt_best-volume-S-for-this-config 0

  ifelse ([a_main-output-is-Service?] of SM-to-update)
  [ set opt_net-profit-of-other-SM best-profit-P-after-last-round
    set opt_best-net-profit-P-for-this-config opt_net-profit-of-other-SM
  ]
  [ set opt_net-profit-of-other-SM best-profit-S-after-last-round
    set opt_best-volume-S-for-this-config opt_net-profit-of-other-SM
  ]
end

to optimise-price-for-one-SM [ SM-to-update other-SM-of-PB horizon capacity-fixed? ] ; PB procedure
  ;; this procedure finds the optimal price for the specified SM based on market research, taking into account possible effects on the profit derived
from the other SM of this PB (if any)

  initialise-opt-variables-for-new-optimisation SM-to-update other-SM-of-PB horizon capacity-fixed?

  ;; ask the consumers in the CA-sample at what price they would switch to SM-to-update, how much they would buy and what their alternative offer is.
  do-market-research

  if (length opt_sorted-market-research-results-nlist) > 5
  [ ;; calculate all additional profit for other-SM-of-PB for the consumers that had SM-to-update as best option and other-SM-of-PB as fallback option
    if opt_consider-other-SM? [ calculate-max-possible-profit-other-SM ]

    find-best-price-based-on-MR-results
  ]

  implement-price-optimisation-results
end

to do-market-research ; PB procedure
  ;; this market research is an essential part of price optimisation and determines for the entire CA-sample their willingness to pay for the offer
being optimised

  let market-research-results-nlist (list)
  ask opt_CA-sample
  [ let market-research-results-list market-research-switch-price-and-volume myself ([a_library-object] of opt_SM-to-update) opt_offer-price-list
opt_horizon
    set market-research-results-nlist lput market-research-results-list market-research-results-nlist
  ]

  ;; sort the results from the market research from high price to low
  set opt_sorted-market-research-results-nlist sort-by [first ?1 > first ?2] market-research-results-nlist

  if g_display-market-research-numbers? [ show (word "tick " current-tick ": market research results: " opt_sorted-market-research-results-nlist) ]

  ;; filter out all results with a price lower than the variable-cost-per-unit: they will never result in a profit
  set opt_sorted-market-research-results-nlist filter [ first ? >= opt_variable-cost-per-unit ] opt_sorted-market-research-results-nlist

  if g_display-market-research-numbers?
  [ show (word "SM to update: " [a_name] of opt_SM-to-update "  other-SM-of-PB: " ifelse-value is-agent? opt_other-SM-of-PB [[a_name] of opt_other-SM-
of-PB] ["none"]) ]

  if g_display-PB-strategic-reconsideration-updates? ;or true
  [ show (word [a_name] of opt_SM-to-update "  number of respondents: " (length opt_sorted-market-research-results-nlist) "  variable costs: "
opt_variable-cost-per-unit) ]
end

to calculate-max-possible-profit-other-SM ;; PB procedure
  ;; this procedure determines how much profit the other SM of this PB would generate if the current SM-to-update would not exist

  foreach opt_sorted-market-research-results-nlist
  [ let results-list ?
    let is-current-customer? item 2 results-list
    let fallback-SM item 3 results-list
    let intermediate-best-SM item 4 results-list
    let output-volume-for-fallback-SM item 5 results-list
    if fallback-SM = opt_other-SM-of-PB and intermediate-best-SM = opt_SM-to-update
    [ let extrapolation-multiplier 1 / (ifelse-value is-current-customer? [a_fraction-of-current-customers-chosen-for-CA-sample] [a_fraction-of-
current-non-customers-chosen-for-CA-sample])
      let volume-for-alternative-SM output-volume-for-fallback-SM * (1 - opt_risk-factor-other-SM) * extrapolation-multiplier

      let associated-MM-output-volume volume-for-alternative-SM * opt_other-SM-input-to-output-ratio
      let additional-profit associated-MM-output-volume * opt_gross-profit-per-MM-output-unit-of-other-SM
      set opt_net-profit-of-other-SM opt_net-profit-of-other-SM + additional-profit

      set opt_cumulative-input-volume-for-other-SM opt_cumulative-input-volume-for-other-SM + associated-MM-output-volume
```

```
      if g_display-market-research-numbers? or display-additional-sim-opt-updates?
      [ show (word "market research results: " results-list "  volume-for-alternative-SM: " volume-for-alternative-SM "  associated-MM-output-volume: "
associated-MM-output-volume "  additional-profit: " additional-profit "  net-profit-of-other-SM: " opt_net-profit-of-other-SM) ]
    ]
  ]
end

to find-best-price-based-on-MR-results ;; PB procedure
  ;; this procedure calculates for each relevant price (based on market research) the expected profit, taking into account the substitution effect on
the other SM of this PB, if any

  if g_display-market-research-numbers?
  [ show (word "  other SM: " opt_other-SM-of-PB "  updated net profit other SM: " opt_net-profit-of-other-SM " other SM profit per MM output unit: "
opt_gross-profit-per-MM-output-unit-of-other-SM)
    show (word "other-SM-i-o-ratio: " opt_other-SM-input-to-output-ratio "  other-SM-gross-MM-profit: " opt_gross-profit-per-MM-output-unit-of-other-SM
"  other-SM-input-volume: " opt_cumulative-input-volume-for-other-SM "  other SM profit: " opt_net-profit-of-other-SM "  free-capacity: " opt_free-
capacity)
  ]

  ;; each point on the 'demand curve' contains the switching price, demand and alternative offer of one consumer in the CA-sample
  foreach opt_sorted-market-research-results-nlist
  [ ;; set price just a little lower than the price in the demand curve, because that's the break-even price
    let price-volume-alternativeSM-formerbestSM ?
    set opt_price 0.9999 * (first price-volume-alternativeSM-formerbestSM)

    if opt_price > opt_max-price [ set opt_price opt_max-price ]

    if opt_price > opt_min-price
    [ let is-current-customer? item 2 price-volume-alternativeSM-formerbestSM
      let extrapolation-multiplier 1 / (ifelse-value is-current-customer? [a_fraction-of-current-customers-chosen-for-CA-sample] [a_fraction-of-
current-non-customers-chosen-for-CA-sample])

      let sales-to-this-customer item 1 price-volume-alternativeSM-formerbestSM * (1 - opt_risk-factor-this-SM) * extrapolation-multiplier

      if opt_consider-other-SM?
      [ let fallback-SM item 3 price-volume-alternativeSM-formerbestSM
        let output-volume-for-fallback-SM item 5 price-volume-alternativeSM-formerbestSM
        evaluate-lost-sales sales-to-this-customer fallback-SM output-volume-for-fallback-SM extrapolation-multiplier
      ]

      ;; calculate how much volume can be sold for the SM-to-update, given the capacity constraint
      set opt_cumulative-input-volume-for-this-SM opt_cumulative-input-volume-for-this-SM + (sales-to-this-customer * opt_this-SM-input-to-output-
ratio)

      ;; in case of strategic reconsideration, it is possible to change the production capacity.
      ;; In that case, calculate below what new capacity level is needed
      ifelse opt_capacity-fixed?
      [ set opt_effective-input-volume min (list opt_free-capacity opt_cumulative-input-volume-for-this-SM) ]
      [ set opt_effective-input-volume opt_cumulative-input-volume-for-this-SM
        set opt_new-total-capacity (opt_effective-input-volume + opt_cumulative-input-volume-for-other-SM) / opt_horizon * a_capacity-over-expected-
sales-ratio

        set opt_new-total-capacity max (list opt_new-total-capacity [a_contracted-output-volume-per-btu] of opt_associated-MM)
      ]

      evaluate-new-results-during-price-optimisation
    ]
  ]
end

to evaluate-lost-sales [ sales-to-this-customer fallback-SM alternative-volume extrapolation-multiplier ] ;; PB procedure
  ;; calculates how much profit of the other SM of this PB will be lost due to a switch from that SM to the SM-to-update.
  ;; But only if the alternative offer was other-SM-of-PB AND that was the best offer regardless of the price of the SM-to-update (otherwise it's no
LOST volume)

  let replaced-input-volume-of-other-SM 0
  let lost-sales 0
  if (fallback-SM = opt_other-SM-of-PB)
  [ set lost-sales alternative-volume * (1 - opt_risk-factor-other-SM) * extrapolation-multiplier
    set replaced-input-volume-of-other-SM lost-sales * opt_other-SM-input-to-output-ratio
    set opt_cumulative-input-volume-for-other-SM opt_cumulative-input-volume-for-other-SM - replaced-input-volume-of-other-SM
  ]

  let additional-lost-profit replaced-input-volume-of-other-SM * opt_gross-profit-per-MM-output-unit-of-other-SM

  set opt_net-profit-of-other-SM opt_net-profit-of-other-SM - additional-lost-profit

  let input-volume-to-this-customer sales-to-this-customer * opt_this-SM-input-to-output-ratio

  if (g_display-market-research-numbers? or display-additional-sim-opt-updates?) and replaced-input-volume-of-other-SM > 0
  [ show (word "sales to this customer: " sales-to-this-customer "  in terms of input volume: " input-volume-to-this-customer"  lost sales: " lost-
sales "  replaced input volume: " replaced-input-volume-of-other-SM "  additional-lost-profit: " additional-lost-profit "  new net profit other SM: "
opt_net-profit-of-other-SM) ]

  ;; calculate how much capacity will be available in total for the SM-to-update, taking into account the possible additional switch away from the
other SM of this PB
  set opt_free-capacity opt_free-capacity + replaced-input-volume-of-other-SM
end

to evaluate-new-results-during-price-optimisation ;; PB procedure
  ;; this procedure calculates for one point on the market-based demand 'curve' the expected profit, and compares it with the best profit found so far

  let new-volume-sold-for-this-SM opt_effective-input-volume / opt_this-SM-input-to-output-ratio

  ;; calculate the total gross profit made for the SM-to-update
  let gross-margin-at-this-price opt_price - opt_variable-cost-per-unit
  let gross-profit-of-this-SM new-volume-sold-for-this-SM * gross-margin-at-this-price
  let net-profit-of-this-SM gross-profit-of-this-SM - [a_structural-costs-per-btu] of opt_SM-to-update
```

```
  let price-per-unit-of-capacity [a_structural-costs-per-unit-of-capacity] of [a_library-object] of opt_associated-MM
  let total-structural-costs-for-MM opt_new-total-capacity * price-per-unit-of-capacity * opt_horizon

  let total-net-profit-for-both-SMs net-profit-of-this-SM + opt_net-profit-of-other-SM - total-structural-costs-for-MM

  ;; if the total gross profit is best overall, save the associated price and the new best gross profit
  if total-net-profit-for-both-SMs > opt_best-profit-for-this-config
  [ set opt_best-capacity-for-this-config opt_new-total-capacity

    ifelse [a_main-output-is-Service?] of opt_SM-to-update
    [ set opt_best-price-P-for-this-config ifelse-value (is-agent? opt_other-SM-of-PB) [ [a_offer-price-per-unit] of opt_other-SM-of-PB ] [ 0 ]
      set opt_best-price-S-for-this-config opt_price
      set opt_best-net-profit-S-for-this-config net-profit-of-this-SM
      set opt_best-net-profit-P-for-this-config opt_net-profit-of-other-SM
      set opt_best-volume-S-for-this-config opt_cumulative-input-volume-for-this-SM
      set opt_best-volume-P-for-this-config opt_cumulative-input-volume-for-other-SM
    ]
    [ set opt_best-price-P-for-this-config opt_price
      set opt_best-price-S-for-this-config ifelse-value (is-agent? opt_other-SM-of-PB) [ [a_offer-price-per-unit] of opt_other-SM-of-PB ] [ 0 ]
      set opt_best-net-profit-P-for-this-config net-profit-of-this-SM
      set opt_best-net-profit-S-for-this-config opt_net-profit-of-other-SM
      set opt_best-volume-P-for-this-config opt_cumulative-input-volume-for-this-SM
      set opt_best-volume-S-for-this-config opt_cumulative-input-volume-for-other-SM
    ]
    set opt_best-profit-for-this-config total-net-profit-for-both-SMs
  ]

  if g_display-market-research-numbers?
  [ show ""
    show (word "tick " current-tick ": market research intermediate results. price (real,min,max): " opt_price " " opt_min-price " " opt_max-price "
free-capacity: " opt_free-capacity "  new capacity: " opt_new-total-capacity)
    show (word "       cum-input-volume: " opt_cumulative-input-volume-for-this-SM "  eff-input-volume:" opt_effective-input-volume "  new-volume-sold:
" new-volume-sold-for-this-SM)
    show (word "       gross margin: " gross-margin-at-this-price "  this-SM-profit: " net-profit-of-this-SM "  other-SM-profit: " opt_net-profit-of-
other-SM "  MM costs: "  total-structural-costs-for-MM "  total profit: " total-net-profit-for-both-SMs)
    show ""
  ]

  if display-additional-sim-opt-updates? [ show (word "  this-SM-profit: " gross-profit-of-this-SM) ]
end

to implement-price-optimisation-results ;; PB procedure
  ;; update the price and target stock level of the SM-to-update based on the optimisation results

  ask opt_SM-to-update
  [ let best-price-found ifelse-value a_main-output-is-Service? [ opt_best-price-S-for-this-config ] [ opt_best-price-P-for-this-config ]
    let best-volume-found ifelse-value a_main-output-is-Service? [ opt_best-volume-S-for-this-config ] [ opt_best-volume-P-for-this-config ]

    set-SM-price self (list 0 best-price-found 0)

    let expected-sales-per-btu best-volume-found / opt_horizon
    if a_main-output-is-Tool? [ update-target-stock-size expected-sales-per-btu ]
  ]
end


to backup-best-offer-properties  ;; CB/C procedure
  ;; save the properties of the currently identified best offer as attributes of this CB/C, allowing to use the a_best-offer-xxx attributes during an
optimisation cycle

  set a_backup-best-offer-SM a_best-offer-SM
  set a_backup-best-offer-costs-per-btu a_best-offer-costs-per-btu
  set a_backup-best-offer-virtual-loyalty-bonus a_best-offer-virtual-loyalty-bonus
  if (breed = Consuming-Businesses) [ set a_backup-best-offer-profit-per-btu a_best-offer-profit-per-btu ]
  set a_backup-best-offer-fit a_best-offer-fit
  set a_backup-best-offer-preference-violation a_best-offer-preference-violation
end

to reset-best-offer-properties-from-backup  ;; CB/C procedure
  ;; restore the best offer properties from the 'backup' attributes to the actual a_best-offer-xxx properties

  update-best-offer-attributes a_backup-best-offer-SM a_backup-best-offer-costs-per-btu a_backup-best-offer-virtual-loyalty-bonus (ifelse-value (breed
= Consuming-Businesses) [a_backup-best-offer-profit-per-btu] [0]) a_backup-best-offer-fit a_backup-best-offer-preference-violation
end


to-report calculate-variable-cost-per-unit-SM [ SM ]  ;; PB procedure
  ;; Calculates the variable-cost-per-unit of one additional output unit of the specified SM

  let SM-variable-cost-per-unit-of-SM-output [calculate-cost-of-secondary-inputs-and-outputs + a_other-variable-costs-per-unit] of SM
  let MM-SM-ratio [ a_main-input-ratio-in-full-unit-equivalents ] of SM
  let MM-variable-cost-per-unit-of-SM-output MM-SM-ratio * [ calculate-cost-of-secondary-inputs-and-outputs + a_other-variable-costs-per-unit] of
[a_associated-MM] of SM

  let total-variable-cost-per-unit SM-variable-cost-per-unit-of-SM-output + MM-variable-cost-per-unit-of-SM-output

  ask SM
  [ if a_main-output-is-Service?
    [ let main-input-disposal-costs a_main-input-ratio-in-full-unit-equivalents * calculate-cost-of-product-disposal a_main-input-id 0  ;; when the
input product is fully consumed, it has 0 use amount left
      set total-variable-cost-per-unit total-variable-cost-per-unit + main-input-disposal-costs
    ]
  ]

  report total-variable-cost-per-unit
end

to produce-output ; PB procedure
  ; input/output conversion at the MM and SM, according to specified production volumes
```

```
  produce-MM-output
  if has-product-SM? [ produce-product-based-SM-output ]

  ; pay structural costs of the adopted models
  let MM-structural-cost [a_structural-costs-per-btu] of a_MM
  let product-based-SM-structural-cost ifelse-value has-product-SM? [ [a_structural-costs-per-btu] of a_product-based-SM ] [ 0 ]
  let service-based-SM-structural-cost ifelse-value has-service-SM? [ [a_structural-costs-per-btu] of a_service-based-SM ] [ 0 ]

  set a_cash-balance a_cash-balance - MM-structural-cost - product-based-SM-structural-cost - service-based-SM-structural-cost
  if g_display-all-cash-mutations-for-one-PB? and who = 67 [ show (word "money spent on structural costs: " (MM-structural-cost + product-based-SM-
structural-cost + service-based-SM-structural-cost)) ]
end

to produce-MM-output  ;; PB procedure
  ;; Produce according to the differences between actual and target stock of both SMs

  let SM-P a_product-based-SM
  let SM-S a_service-based-SM

  let requested-volume-P ifelse-value is-agent? SM-P [ max (list 0 ([(a_target-stock-size - (stock-size a_main-output-stock)) / a_main-input-ratio-in-
full-unit-equivalents] of SM-P)) ] [ 0 ]  ;; if the target stock size is below the current stock size, just set the requested volume to 0
  let requested-volume-S ifelse-value is-agent? SM-S [ max (list 0 ([(a_target-stock-size - (stock-size a_main-input-stock)) ] of SM-S)) ] [ 0 ]

  let total-requested-volume requested-volume-P + requested-volume-S

  ;; round down: you can never produce more than the capacity allows
  let production-volume floor [a_output-capacity-per-btu + a_overflow-output-capacity-due-to-rounding] of a_MM

  ;; check how much you rounded down: this overflow capacity can be used in the next tick
  ask a_MM [ set a_overflow-output-capacity-due-to-rounding (a_output-capacity-per-btu + a_overflow-output-capacity-due-to-rounding) - production-
volume ]

  ;; determine the actual restock volumes (which are different in case of capacity shortage)
  let volume-P requested-volume-P
  let volume-S requested-volume-S

  if production-volume < total-requested-volume
  [ let allocated-volumes allocated-production-volumes-with-scarcity SM-P SM-S volume-P volume-S requested-volume-P requested-volume-S production-
volume
    set volume-P first allocated-volumes
    set volume-S last allocated-volumes
  ]

  set production-volume volume-P + volume-S

  ;; produce!
  ask a_MM
  [ buy-and-use-secondary-inputs production-volume

    create-and-dispose-secondary-outputs production-volume

    ;; make the specified amount of the main output product
    make-x-outputs production-volume

    ;; pay any applicable variable costs other than those from buying inputs
    let variable-costs production-volume * a_other-variable-costs-per-unit  ;; this refers to all costs of the MM other than those required for buying
inputs/infrastructure/skills/...  i.e. everything that does not have a price in the model
    ask a_owner
    [ set a_cash-balance a_cash-balance - variable-costs
      if g_display-all-cash-mutations-for-one-PB? and who = 67 [ show (word "money spent on variable MM costs: " variable-costs " (" production-volume
" units produced)" ) ]
    ]

    if is-agent? SM-P
    [ if volume-P > stock-size a_main-output-stock [ show (word "production-volume: " production-volume "  volume-P: " volume-P "  volume-S: " volume-
S) ]
      shift-inputs-to-SM SM-P volume-P ]
    if is-agent? SM-S
    [ if volume-S > stock-size a_main-output-stock [ show (word "production-volume: " production-volume "  volume-P: " volume-P "  volume-S: " volume-
S) ]
      shift-inputs-to-SM SM-S volume-S ]
  ]
end

to-report allocated-production-volumes-with-scarcity [ SM-P SM-S volume-P volume-S requested-volume-P requested-volume-S production-volume ]  ;;
generic procedure
  ;; if the capacity is not sufficient, determine how much of the effective production should be allocated to each of the SMs.
  ;; If there is one SM, this is easy: just reduce the volume for that SM.
  ;; If there are two SMs, divide between them based on relative (compared to target stock) total production shortage since last strategic
reconsideration

  ;; check if the PB has two SMs
  ifelse (is-agent? SM-P and is-agent? SM-S)
  [ ;; initialise the allocation procedure by first allocating as much available production as possible or needed to the product-based SM
    set volume-P min (list requested-volume-P production-volume)
    set volume-S production-volume - requested-volume-P

    let target-P [a_target-stock-size] of SM-P
    let target-S [a_target-stock-size] of SM-S

    ;; calculate total shortage of both SMs: the total shortage so far plus the current gap between target and actual stock size
    let shortage-P [a_total-production-shortage] of SM-P + requested-volume-P
    let shortage-S [a_total-production-shortage] of SM-S + requested-volume-S

    ;; based on the initialisation (all to product-based SM), calculate the absolute value of the difference between the relative total shortage of
both SMs.
    ;; The relative total shortage is here the sum of all shortages since the last strategic reconsideration, relative to the target size of the stock.
    let implied-relative-total-shortage-P ((shortage-P - volume-P) / target-P)
```

```
    let implied-relative-total-shortage-S ((shortage-S - volume-S) / target-S)

    let difference abs ( implied-relative-total-shortage-P - implied-relative-total-shortage-S )
    let best-difference-found difference
    let best-volume-found? false

    ;; now incrementally allocate one additional unit from the product-based-SM to the service-based SM, until the difference in relative shortage is
minimised.
    while [ volume-P > 0 and (not best-volume-found?) ]
    [ set volume-P volume-P - 1
      set volume-S volume-S + 1
      set difference abs (((shortage-P - volume-P) / target-P)  -  ((shortage-S - volume-S) / target-S))
      if difference >= best-difference-found
      [ set best-volume-found? true ]
    ]

    set volume-S min (list requested-volume-S production-volume)
    ; if requested-volume-S > production-volume [ set volume-S production-volume show (word current-tick " capacity shortage") ]
    set volume-P production-volume - volume-S

    ;; update total production shortage of both SMs
    ask SM-P [ set a_total-production-shortage a_total-production-shortage + requested-volume-P - volume-P ]
    ask SM-S [ set a_total-production-shortage a_total-production-shortage + requested-volume-S - volume-S ]
  ]
  [ ;; if the PB has just one SM, determine if it's product-based or service-based. Then reduce the allocated volume based on the available production
volume
    ifelse is-agent? SM-P
    [ set volume-P production-volume
      set volume-S 0
      ask a_product-based-SM [ set a_total-production-shortage a_total-production-shortage + requested-volume-P - volume-P ]
    ]
    [ set volume-S production-volume
      set volume-P 0
      ask a_service-based-SM [ set a_total-production-shortage a_total-production-shortage + requested-volume-S - volume-S ]
    ]
  ]

  report (list volume-P volume-S)
end

to buy-and-use-secondary-inputs [ output-volume ]  ;; MM/SM/CM procedure
  ;; this is a step in the production process

  let next-input-idx 0  ;; idx = index
  let n-inputs length a_secondary-input-IDs-list

  ;; traverse all specified inputs and buy them from the world market
  while [ next-input-idx < n-inputs ]
  [ let input-ID item next-input-idx a_secondary-input-IDs-list
    let volume output-volume * item next-input-idx a_secondary-input-ratios-list

    ;; buy from the World Market, pay for it and then directly consume it (because it's used for the production of something else)
    consume-from-world-market input-ID volume

    set next-input-idx next-input-idx + 1
  ]
end

to create-and-dispose-secondary-outputs [ output-volume ]  ;; MM/SM/CM procedure
  ;; this is a step in the production process

  let next-output-idx 0  ;; idx = index
  let n-outputs length a_secondary-output-IDs-list

  ;; traverse all specified outputs and buy them from the world market
  while [ next-output-idx < n-outputs ]
  [ let output-ID item next-output-idx a_secondary-output-IDs-list
    let volume output-volume * item next-output-idx a_secondary-output-ratios-list

    ;; buy from the World Market, pay for it and then directly consume it (because it's used for the production of something else)
    let disposal-cost 0
    ifelse is-a-Tool? output-ID
    [ set disposal-cost volume * calculate-cost-of-product-disposal output-ID 1 ]
    [ set disposal-cost calculate-cost-of-product-disposal output-ID volume ]

    ask a_owner
    [ set a_cash-balance a_cash-balance - disposal-cost
      if g_display-all-cash-mutations-for-one-PB? and who = 67 [ show (word "money spent on disposal of secondary outputs: " disposal-cost " (" output-
volume " units produced by " myself ")" ) ]
    ]

    set next-output-idx next-output-idx + 1
  ]
end


to produce-product-based-SM-output  ;; PB procedure
  ;; this procedure shifts products produced by the MM to the product-based SM, and then converts SM inputs to SM outputs

  ask a_product-based-SM
  [ ;; consume main input
    let converted-amount-of-main-input stock-size a_main-input-stock  ;; produce based on what was allocated to the input stock of this SM after
production by the MM
    let output-volume-today converted-amount-of-main-input / a_main-input-ratio-in-full-unit-equivalents

    convert-main-input-SM converted-amount-of-main-input
    buy-and-use-secondary-inputs output-volume-today
    create-and-dispose-secondary-outputs output-volume-today
```

```
    ;; make the specified amount of the main output product
    make-x-outputs output-volume-today
    set a_number-produced a_number-produced + output-volume-today

    ;; pay any applicable variable costs other than those from buying inputs
    let variable-costs output-volume-today * a_other-variable-costs-per-unit
    ask a_owner
    [ set a_cash-balance a_cash-balance - variable-costs
      if g_display-all-cash-mutations-for-one-PB? and who = 67 [ show (word "money spent on variable SM costs: " variable-costs " (" output-volume-
today " units produced)" ) ]
    ]
  ]
end

to shift-inputs-to-SM [ SM volume ]  ;; MM procedure
  ;; move the specified volume of products from the output stock of an MM to the input stock of the calling SM

  ifelse a_main-output-is-Tool?
  ;; in case of Tools-based, shift the number of Tools one by one from the MM output stock to the SM input stock (first in, first out)
  [ repeat volume
    [ let shifted-product min-one-of a_main-output-stock [who]  ;; the product with the lowest 'who' is the one that was produced first of all of
products currently in stock
      set a_main-output-stock remove-from-turtle-set shifted-product a_main-output-stock

      ;; depending on the SM that it goes to, the same product may have a different lifetime (e.g. owning vs renting)
      ask shifted-product
      [ set a_remaining-use-time a_remaining-use-time * [a_use-time-multiplier-for-main-input] of SM
        visualise-product SM true
      ]

      ask SM [ set a_main-input-stock add-to-turtle-set shifted-product a_main-input-stock ]
    ]
  ]

  ;; in case of Consumable-based, decrease the quantity of the Consumable in the MM output stock and increase the quantity of the Consumable in the SM
input stock by the specified amount
  [ ask a_main-output-stock [ set a_quantity a_quantity - volume ]

    ask SM [ ask a_main-input-stock [ set a_quantity a_quantity + volume ] ]
  ]
end

to make-x-outputs [ x ] ;; MM or SM procedure
  ;; create the specified number of outputs as new objects (or in case of Consumable-based output, increase the quantity of the Consumable object
representing the stock)

  ifelse a_main-output-is-Tool?
  [ repeat x [ let new-tool make-one-tool ] ]
  [ ask a_main-output-stock [ set a_quantity a_quantity + x ] ]
end

to-report make-one-tool  ;; MM or SM procedure
  ;; this procedures creates one Tool as a new object, initialises its attributes, and adjusts the relevant stock

  let model self
  let new-product nobody

  ask a_main-output-library-object
  [ hatch 1
    [ set breed ifelse-value (breed = Tools-lib) [ Tools ] [ Consumables ]
      set a_library-object myself
      set a_remaining-use-time [a_total-use-time] of myself
      ask model [ set a_main-output-stock add-to-turtle-set myself a_main-output-stock ]
      visualise-product model false
      set new-product self
    ]
  ]
  report new-product
end

to-report make-initial-empty-Consumable [ for-input? ]  ;; MM/SM/CM procedure
  ;; this procedure creates an initial Consumable stock with quantity 0. This is needed because the size of a stock of a Consumable is stored through
the 'a_quantity' attribute of a single Consumable
  ;; the procedure can be used either for the input stock ('for-input?' = true) or output stock ('for-input?' = false)

  let model self
  let consumable nobody

  ifelse for-input?
  [ ask a_main-input-library-object
    [ hatch-Consumables 1
      [ set a_library-object myself
        ask model
        [ ifelse breed = Consumption-Models
          [ set a_associated-owned-products (turtle-set myself) ]
          [ set a_main-input-stock myself ]
        ]
        visualise-product model true
        set consumable self
      ]
    ]
  ]
  [ ask a_main-output-library-object
    [ hatch-Consumables 1
      [ set a_library-object myself
        ask model [ set a_main-output-stock (turtle-set myself) ]
        visualise-product model false
        set consumable self
      ]
```

```
    ]
  ]
  report consumable
end

to satisfy-need ; CB/C procedure
  ;; this procedure makes the consuming agent consume all the relevant inputs to consume her daily amount of Need

  let Need-amount a_Need-per-btu
  ask a_Consumption-Model
  [ let next-input-idx 0
    let n-inputs length a_secondary-input-IDs-list

    ;; consume main input
    let needed-amount-of-main-input Need-amount * a_main-input-ratio
    consume-main-input-CM needed-amount-of-main-input

    buy-and-use-secondary-inputs Need-amount
    create-and-dispose-secondary-outputs Need-amount
  ]

  ;; pay the structural costs associated with the chosen CM
  set a_cash-balance a_cash-balance - [a_structural-costs-per-btu] of a_Consumption-Model

  ;; keep track of the total time spent on consumption by this agent
  set a_time-spent [a_labour-time] of a_Consumption-Model * a_Need-per-btu
  set a_total-time-spent a_total-time-spent + a_time-spent

  ;; if consuming business: pay fixed cost and collect revenue
  if breed = Consuming-Businesses
  [ set a_cash-balance a_cash-balance - a_default-cost-per-btu + a_default-revenue-per-btu * [a_revenue-multiplier] of a_Consumption-Model ]
end

to convert-main-input-SM [ amount ]  ;; SM procedure
  ;; This procedure consumes the specified 'amount' from the SM's input stock you need to use for production during this tick. Completely used-up
products are removed from the model

  ;; if the main input is a Tool:
  ifelse a_main-input-is-Tool?
  [ let remaining-amount-to-consume amount
    ;; it can be that the SM currently has multiple Tools in her input stock. In that case, traverse them one by one until the full amount needed is
consumed
    while [ remaining-amount-to-consume > 0 ]
    [ ask min-one-of a_main-input-stock [ who ]  ;; always use up the oldest Tool first
      [ ifelse [a_main-output-is-Service?] of myself
        ;; if the main output is a service, 'amount' is specified in amount of use time of the input Tool
        [ ifelse a_remaining-use-time >= remaining-amount-to-consume
          ;; if the current product still has sufficient use time, just use that product and update its remaining use-time accordingly
          [ set a_remaining-use-time a_remaining-use-time - remaining-amount-to-consume
            set remaining-amount-to-consume 0
          ]

          ;; if we need all (and maybe more) remaining use time of the selected product, use all that it has left, update the remaining amount to be
consumed and remove the product from the model
          [ set remaining-amount-to-consume remaining-amount-to-consume - a_remaining-use-time
            set a_remaining-use-time 0
            die
          ]
        ]
        ; if the main output is not a service, 'amount' is specified in number of Tools and they all need to disappear in the production process
        [ set remaining-amount-to-consume remaining-amount-to-consume - 1
          die   ;; tools being used in the production process are just removed from the model
        ]
      ]
    ]
  ]

  ;; if the main input is a Consumable, just decrease the quantity of the Consumable object representing the input stock:
  [ ask a_main-input-stock
    [ if a_quantity < amount [ show (word "tick " current-tick ": warning! Consumer " [who] of a_owner " has insufficient amount of the input
Consumable for SM " who ". This should not be possible!") ]
      set a_quantity a_quantity - amount
    ]
  ]
end

to consume-main-input-CM [ amount ]  ;; CM procedure
  ;; this procedure just branches off to either the 'consume-service' subprocedure or the 'consume-product-CM' subprocedure, depending on the type of
input

  ifelse a_main-input-is-Service?
  [ consume-service amount ]
  [ consume-product-CM amount ]
end

to consume-service [ amount ]  ;; CM procedure
  ;; consuming a service means asking the seller to deliver the required amount of Service. This directly satisfies the Need of the consuming agent

  ask a_owner
  [ ask [a_seller] of a_service-contract
    [ deliver-service amount ]

    pay-seller-for-unit amount
    pay-seller-time-based-fee
  ]
end

to deliver-service [ amount ]  ;; PB procedure
```

```
  ;; delivering a Service means a conversion of SM inputs to SM outputs, among which the Service, although the Service is not actually created as an
object (it is assumed to be directly consumed)

  ask a_service-based-SM
  [ let next-input-idx 0
    let n-inputs length a_secondary-input-IDs-list

    ;; consume main input
    let needed-amount-of-main-input amount * a_main-input-ratio
    convert-main-input-SM needed-amount-of-main-input

    ;; traverse all specified secondary inputs
    while [ next-input-idx < n-inputs ]
    [ let input-ID item next-input-idx a_secondary-input-IDs-list
      let input-ratio amount * item next-input-idx a_secondary-input-ratios-list

      consume-from-world-market input-ID amount

      set next-input-idx next-input-idx + 1
    ]
  ]
end

to consume-product-CM [ amount ]  ;; CM procedure. 'amount' is the amount of the main input product that you need to consume here
  ;; This procedure consumes the specified 'amount' from the associated owned product(s) in order to satisfy the agent's Need during this tick.
Completely used-up products are removed from the model

  ;; check if the CB/C has sufficient products to be consumed (will return an error if she tries to use a nonexisting product)
  ifelse [a_needed-n-new-products] of a_owner = 0  ;; 'a_needed-n-new-products' stores whether the CB/C still needs to buy new products (because she
ran out of them), and if yes, how many.
  [ let needed-n-new-products 0

    ifelse a_main-input-is-Tool?
    ;; if the CM is based on a tool:
    [ let remaining-amount-to-consume amount

      ;; it can be that the CM currently has multiple associated owned Tools (e.g. because one is almost used up). In that case, traverse them one by
one until the full amount needed is consumed
      while [ remaining-amount-to-consume > 0 ]
      [ ask most-used-associated-owned-product  ;; always use up the Tool with the lowest remaining lifetime first
        [ ifelse a_remaining-use-time >= remaining-amount-to-consume
          ;; if the current product still has sufficient use time, just use that product and update its remaining use-time accordingly
          [ set a_remaining-use-time a_remaining-use-time - remaining-amount-to-consume
            set remaining-amount-to-consume 0
            set needed-n-new-products ceiling ([a_consumed-main-input-volume-per-btu] of myself - (a_remaining-use-time / [a_total-use-time] of
a_library-object))  ;; this is always the last product used, so here we calculated the number of products needed in the next tick. total use time of a
consumer-owned product should always be equal to that of the library object (use time multipliers should only apply to service-based Tools)
          ]

          ;; if we need all (and maybe more) remaining use time of the selected product, use all that it has left, update the remaining amount to be
consumed and remove the product from the model
          [ set remaining-amount-to-consume remaining-amount-to-consume - a_remaining-use-time
            set a_remaining-use-time 0
            set needed-n-new-products ceiling [a_consumed-main-input-volume-per-btu] of myself  ;; this is for the case where the remaining amount was
just enough to satisfy the Need during this tick
            die
          ]
        ]
      ]
    ]
    ;; if the main input is a Consumable, just decrease the quantity of the Consumable object representing the input stock:
    [ ask a_associated-owned-products [ set a_quantity a_quantity - amount ]
      set needed-n-new-products a_consumed-main-input-volume-per-btu - sum [a_quantity] of a_associated-owned-products
    ]

    ;; make sure the owner knows how many new products she has to restock
    ask a_owner
    [ set a_needed-n-new-products needed-n-new-products ]
  ]
  [ show (word "tick " current-tick ": Warning! Consuming agent " [who] of a_owner " does not possess the required product!") ]

  ;; if 'a_automatic-repurchase?' of the CM is false, the fact that the current product is fully used up triggers the agent to reconsider her
consumption strategy
  ;; in case of Consumables, 'a_automatic-repurchase?' should always be 'true' in our current assumptions
  ask a_owner
  [ if [not a_automatic-repurchase?] of a_Consumption-Model and (a_needed-n-new-products > 0)
    [ set a_time-of-next-strategic-reconsideration current-tick + 1  ]
  ]
end


to-report calculate-preference-fit [ model ]  ;; PB or CB/C procedure.
  ;; calculates the 'fit' between the scores of the specified model and the Preferences of the calling agent. Returns a value between 1 and 10

  ;; The scores-list is the list of scores on all the types of Preferences for the model you want to evaluate
  let scores-list [a_preference-scores-list] of model

  ; add market share score to the list of preference scores
  let relevant-item nobody
  ask model [ set relevant-item ifelse-value ((breed = Consumption-Models) or (breed = Consumption-Models-lib)) [ a_main-input-library-object ] [
a_main-output-library-object ] ]
  let market-share-score [a_market-share-score] of relevant-item
  set scores-list lput market-share-score scores-list

  let combined-score sum (map [ ?1 * ?2 ] scores-list a_preference-weights-list)

  ; normalise by dividing the total weighted score by the sum of scores. This ensures that the procedure returns a normalised value between 1 and 10:
  let sum-of-weights sum a_preference-weights-list
  report combined-score / sum-of-weights
```

```
end


to-report calculate-max-preference-threshold-violation [ model for-CM? ]   ;; PB or CB/C procedure
  ;; compares the scores of the specified model with the thresholds of the calling agent. Returns the maximum relative threshold violation, e.g. 0.6
for a 60% violation

  let scores-list [a_preference-scores-list] of model

  ; add market share score to the list of preference scores
  let relevant-item nobody
  ask model [ set relevant-item ifelse-value for-CM? [ a_main-input-library-object ] [ a_main-output-library-object ] ]
  let market-share-score [a_market-share-score] of relevant-item
  set scores-list lput market-share-score scores-list

  let min-threshold-violations (map [ ifelse-value ((?1 > 0) and (?2 > ?1)) [ (?2 - ?1) / ?1 ] [ 0 ] ] a_preference-min-thresholds-list scores-list)

  report max min-threshold-violations
end

to consume-from-world-market [ input-ID amount ]   ;; MM or SM or CM procedure
  ;; currently this procedure just adjusts the cash balance of the owner of the calling agent, representing a purchase

  let price World-Market-offered-price input-id

  ask a_owner
  [ set a_cash-balance a_cash-balance - price * amount
    if g_display-all-cash-mutations-for-one-PB? and who = 67 [ show (word "money spent on consumption from world market: " (price * amount) " (" amount
" units produced)" ) ]
  ]
end

to restock-products  ;; CB/C procedure
  ;; procedure where a consuming agent buys new products based on the calculated amount during the 'satisfy-Need' procedure. Does not apply to service-
based CM's (they have no stock at the CB/C)

  let failure? false
  ifelse [a_main-input-is-Tool?] of a_Consumption-Model
  ; if the main input is a Tool, restock the products you need, one-by-one
  [ while [ (not failure?) and a_needed-n-new-products > 0 ]
    [ ;; check that the needed Tool is available in the supplier's stock
      let stock [a_main-output-stock] of a_current-suppliers-SM

      ;; if in stock, pay the seller and acquire the Tool as specified in that procedure
      ifelse stock-size stock > 0
      [ let sold-product min-one-of stock [who]
        pay-seller-for-unit 1
        acquire-Tool-for-Consumption sold-product
        set a_needed-n-new-products a_needed-n-new-products - 1
      ]

      ;; if not in stock, update the 'failure?' variable to make sure a strategic reconsideration will be triggered
      [ set failure? true
        show (word "tick " current-tick ": Sales Model " [who] of a_current-suppliers-SM " out of stock!")
      ]
    ]
  ]
  ; if the main input is a Consumable, directly adjust the 'a_quantity' of the owned Consumable object
  [ if [a_main-input-is-Consumable?] of a_Consumption-Model
    [ ;; check that the required amount of the Consumable is available in the supplier's stock
      let stock [a_main-output-stock] of a_current-suppliers-SM
      ;; if in stock, pay the seller and acquire amount of the Consumable as specified in that procedure
      ifelse (stock-size stock) >= a_needed-n-new-products
      [ pay-seller-for-unit a_needed-n-new-products
        acquire-Consumable-for-Consumption a_needed-n-new-products
        set a_needed-n-new-products 0
      ]
      ;; if not in stock, update the 'failure?' variable to make sure a strategic reconsideration will be triggered
      [ set failure? true
        show (word "tick " current-tick ": Sales Model " [who] of a_current-suppliers-SM " out of stock!")
      ]
    ]
  ]
  ;; if the restocking was not successful, the agent has to select a new CM. Make sure that she does this by updating the 'a_time-of-next-strategic-
reconsideration' attribute
  if failure?
  [ set a_time-of-next-strategic-reconsideration current-tick + 1 ]
end

to pay-seller-one-time-fee  ;; CB/C procedure
  ;; adjust the cash balances of both the calling consuming agent and the owner of her supplier, based on the initial one-time fee of the Service
Contract

  let fee [a_offer-price-one-time-contract-cost] of a_current-suppliers-SM
  set a_cash-balance a_cash-balance - fee
  ask a_current-supplier
  [ set a_cash-balance a_cash-balance + fee
    if g_display-all-cash-mutations-for-one-PB? and who = 67 [ show (word "one time fee received: " fee) ]
  ]
end

to pay-seller-for-unit [ amount ]  ;; CB/C procedure
  ;; adjust the cash balances of both the calling consuming agent and the owner of her supplier

  let price [a_offer-price-per-unit] of a_current-suppliers-SM
  let total-price price * amount
  set a_cash-balance a_cash-balance - total-price
  ask a_current-supplier
  [ set a_cash-balance a_cash-balance + total-price
```

```
    if g_display-all-cash-mutations-for-one-PB? and who = 67 [ show (word amount "  units sold for a total of " total-price) ]
  ]

  ask a_current-suppliers-SM [ set a_number-sold a_number-sold + amount ]
end

to pay-seller-time-based-fee  ;; CB/C procedure
  ;; adjust the cash balances of both the calling consuming agent and the owner of her supplier, based on the periodic costs associated with the
Service Contract

  let fee [a_offer-price-per-btu] of a_current-suppliers-SM
  set a_cash-balance a_cash-balance - fee
  ask [a_owner] of a_current-suppliers-SM
  [ set a_cash-balance a_cash-balance + fee
    if g_display-all-cash-mutations-for-one-PB? and who = 67 [ show (word "  time-based fee received: " fee) ]
  ]

end

to update-market-shares  ;; observer procedure
  ;; recalculates the relative (0-1) market shares (in terms of fraction of total Need in the model satisfied through that type) of all sellable
product and service types specified in the model

  let total-need-per-btu sum [a_Need-per-btu] of g_all-consuming-agents
  ;; traverse the sellable products and services one by one
  ask g_all-sellable-products-and-services-lib
  [ let product-or-service-id a_library-id
    ;; find out which consumers rely on this product or Service to satisfy their Need
    let agents-that-use-this g_all-consuming-agents with [ [a_main-input-id] of a_Consumption-Model = product-or-service-id ]
    ;; sum the total Need per btu of those agents
    let total-Need-for-this-product-or-service sum [a_Need-per-btu] of agents-that-use-this
    ;; normalise based on the total amount of Need in the model
    set a_market-share total-Need-for-this-product-or-service / total-need-per-btu
  ]

  calculate-market-share-scores
  display-market-shares
end

to calculate-market-share-scores  ;; observer procedure
  ;; the market share scores are calculated relative to the largest market share. E.g. if the largest market share is 50%, a product with a share of
22% gets a score of 22%/50% * 10 = 4.4

  ;; find the biggest market share
  let max-market-share max [a_market-share] of g_all-sellable-products-and-services-lib
  ;; one by one, calculate the scores of all sellable product and service types
  ask g_all-sellable-products-and-services-lib
  [ let relative-market-share a_market-share / max-market-share
    set a_market-share-score ceiling (5 * relative-market-share)  ;; the market share scores run from 0 to 5
  ]

  ;; market penetration scores are always rounded up, so that there's a difference in scores for products with 0 relative market penetration (compared
to largest) and 0.01
end

to display-market-shares  ;; observer procedure
  ;; update the visual representation of the market shares

  let sorted-products-and-services-list sort-on [a_library-id] g_all-sellable-products-and-services-lib

  graphics:set-font "serif" "plain" 11
  let share-x 99
  let text-y 74

  ;; clear the market share area in the model dashboard
  graphics:set-fill-color [ 255 255 255 ]
  graphics:fill-rectangle  95 76 4 75

  ;; calculate and display the market shares of the sellable products and services in the model
  foreach sorted-products-and-services-list
  [ ;; display the market share
    let market-share-in-percentage (round (([a_market-share] of ?) * 1000)) / 10
    let text-to-display ifelse-value ((market-share-in-percentage mod 1) = 0) [ (word market-share-in-percentage ".0%") ] [ (word market-share-in-
percentage "%") ]
    graphics:draw-text share-x text-y "E" text-to-display  ;; "E" means align east (i.e. to the right)

    set text-y text-y - 2
  ]
end


to update-resource-LCA-scores ; observer procedure
  ;; determine the total use per resource and disposal per waste based on the current choices of PBs and CB/Cs

  update-CM-adoption-statistics
  update-model-library-objects-LCA-scores

  ask Resources-lib
  [ let index a_library-id - 1
    set a_total-amount-used-as-input-this-time-step sum [a_total-Need-satisfied-through-this-CM * (item index a_LCA-resources-use-per-unit-of-output-
list)] of Consumption-Models-lib
    set a_total-amount-caused-as-waste-this-time-step sum [a_total-Need-satisfied-through-this-CM * (item index a_LCA-associated-wastes-per-unit-of-
output-list)] of Consumption-Models-lib
  ]
end

to update-CM-adoption-statistics ; observer procedure
  ;; make sure the CM-libs 'know' how much they are used per time step
```

```
  ask Consumption-Models-lib
  [ let CM-lib-id a_library-id
    set a_total-Need-satisfied-through-this-CM sum [a_Need-per-btu] of g_all-consuming-agents with [[a_library-id] of a_Consumption-Model = CM-lib-id]
  ]
end

to update-model-library-objects-LCA-scores; observer procedure
  ;; calculate for each library CM the total amount of resources used and wastes disposed per time step, based on input/output conversions at MM, SM
and CM

  ask Manufacturing-Models-lib
  [ let empty-resource-or-waste-list n-values g_number-of-resources [0]
    set a_LCA-resources-use-per-unit-of-output-list empty-resource-or-waste-list
    set a_LCA-associated-wastes-per-unit-of-output-list empty-resource-or-waste-list

    add-LCA-resource-use a_secondary-input-IDs-list a_secondary-input-ratios-list
    add-LCA-associated-wastes a_secondary-input-IDs-list a_secondary-input-ratios-list true
    add-LCA-associated-wastes a_secondary-output-IDs-list a_secondary-output-ratios-list false
  ]

  ask Sales-Models-lib
  [ set a_LCA-resources-use-per-unit-of-output-list map [ ? * a_main-input-ratio-in-full-unit-equivalents ] ([a_LCA-resources-use-per-unit-of-output-
list] of a_associated-MM-lib)
    set a_LCA-associated-wastes-per-unit-of-output-list map [ ? * a_main-input-ratio-in-full-unit-equivalents ] ([a_LCA-associated-wastes-per-unit-of-
output-list] of a_associated-MM-lib)

    add-LCA-resource-use a_secondary-input-IDs-list a_secondary-input-ratios-list
    add-LCA-associated-wastes a_secondary-input-IDs-list a_secondary-input-ratios-list true
    add-LCA-associated-wastes a_secondary-output-IDs-list a_secondary-output-ratios-list false
  ]

  ask Consumption-Models-lib
  [ set a_LCA-resources-use-per-unit-of-output-list map [ ? * a_main-input-ratio-in-full-unit-equivalents ] ([a_LCA-resources-use-per-unit-of-output-
list] of a_associated-SM-lib)
    set a_LCA-associated-wastes-per-unit-of-output-list map [ ? * a_main-input-ratio-in-full-unit-equivalents ] ([a_LCA-associated-wastes-per-unit-of-
output-list] of a_associated-SM-lib)

    add-LCA-resource-use a_secondary-input-IDs-list a_secondary-input-ratios-list
    add-LCA-associated-wastes a_secondary-input-IDs-list a_secondary-input-ratios-list true
    add-LCA-associated-wastes a_secondary-output-IDs-list a_secondary-output-ratios-list false
  ]
end

to add-LCA-resource-use [ material-ids-list material-ratios-list ] ; {MM/SM/CM}-lib  procedure
  ;; supportive procedure to add up LCA scores

  foreach list-indices material-ids-list
  [ let library-id item ? material-ids-list
    let relative-use item ? material-ratios-list
    let resources-use-list ([a_use-of-resources] of library-object-with-id library-id)
    let weighted-resources-use-list map [? * relative-use] resources-use-list

    set a_LCA-resources-use-per-unit-of-output-list (map [?1 + ?2] a_LCA-resources-use-per-unit-of-output-list weighted-resources-use-list)
  ]
end

to add-LCA-associated-wastes [ material-ids-list material-ratios-list materials-are-inputs? ] ; {MM/SM/CM}-lib  procedure
  ;; supportive procedure to add up LCA scores

  foreach list-indices material-ids-list
  [ let library-id item ? material-ids-list
    let relative-use item ? material-ratios-list
    let associated-wastes-list [ ifelse-value materials-are-inputs? [a_associated-wastes] [a_use-of-resources] ] of library-object-with-id library-id
; in case of inputs, add the associated wastes so far. In case of outputs, add the resource use as additional wastes
    let weighted-associated-wastes-list map [? * relative-use] associated-wastes-list

    set a_LCA-associated-wastes-per-unit-of-output-list (map [?1 + ?2] a_LCA-associated-wastes-per-unit-of-output-list weighted-associated-wastes-list)
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;                                            ;;;;;
;;;;;                 Supportive                 ;;;;;
;;;;;                                            ;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

to skip-lines [n-lines-to-skip]  ;; observer procedure
  ;; this is simply an ancillary procedure to skip n lines in a text file. Assumes that a file is 'open' in NetLogo

  repeat n-lines-to-skip
  [ set g_last-scanned-line file-read-line ]
end

to skip-file-to-line [ line ]  ;; observer procedure
  ;; skips in the open text file to the specified line. The line must be exactly the same as what's in the text file: partial matches don't work

  let nextline ""
  while [ not (nextline = line) ]
  [ set nextline file-read-line ]
end

to add-myself-to-library-object-list  ;; library agent procedure
  ;; this procedure makes sure that the calling agent becomes part of the 'g_all-library-objects-sorted-list' global variable

  while [ length g_all-library-objects-sorted-list < a_library-id ]
  [ set g_all-library-objects-sorted-list lput 0 g_all-library-objects-sorted-list ]
```

```
  set g_all-library-objects-sorted-list lput self g_all-library-objects-sorted-list
end

to visualise-product [ model input? ]  ;; TP/CP procedure
  ;; moves the calling product to the appropriate place in the model interface, close to its owner
  ;; 'model' is the MM/SM/CM that the product belongs to, 'input' says whether it is the main input or the main output for that model

  move-to model  show-turtle

  ifelse [breed] of model = Consumption-Models
  [ set heading 0   fd 0.7   set heading 90   bk 0.3 set size 0.5 ]
  [ set heading 90   fd ifelse-value input? [ -0.5 ] [ 0.5 ] set size 0.5 ]
end

to visualise-model  ;; MM/SM/CM procedure
  ;; moves the calling product to the appropriate place in the model interface, close to its owner

  move-to a_owner
  show-turtle

  ;; MMs are placed centrally-left (relative to the owner)
  ifelse breed = Manufacturing-Models
  [ set heading 180   fd 0.7   set heading 90   bk 0.5   set color violet ]

  ;; SMs are placed to the right of the MM, either slightly higher (if product-based) or slightly lower (if service-based)
  [ ifelse breed = Sales-Models
    [ ifelse a_main-output-is-Service?
      [ set heading 180   fd 1.1   set heading 90   fd 0.5   set color last [a_SM-colors] of a_owner ]
      [ set heading 180   fd 0.3   set heading 90   fd 0.5   set color first [a_SM-colors] of a_owner ]
    ]
    ;; CMs are placed centrally (relative to the owner)
    [ ifelse breed = Consumption-Models
      [ set heading 180   fd 0.2   set heading 90 ]
      [ error "No model of this type known!" ]
    ]
  ]
end

to-report next-line-in-file  ;; observer procedure
  ;; this just makes the calling procedure more readable ('next-line-in-file' is easier to understand than 'file-read-line')

  report file-read-line
end

to-report remaining-amount ;; TP/CP procedure
  ;; this reporter reports the relative remaining amount of the product, which can be either relative use time (if the product is a Tool) or quantity
(if the product is a Consumable)

  report ifelse-value (breed = Tools) [ a_remaining-use-time / [a_total-use-time] of a_library-object ] [ a_quantity ]
end

to-report get-library-id [ type-id indiv-id ]   ;; generic procedure
  ;; the library ids of all agents and objects in the model are all unique, so there is a big list of unique IDs that represent a specific agent or
object in the model
  ;; this procedure reports this overall/global library-id based on the object type (id) and the element id within this object group.
  ;; it is needed during initialisation of library-objects but also at other places
  ;; for example, PBs have type-id 1, CBs have type-id 2. If there are 2 PBs, they get library-ids 1 and 2 and the first CB will have library-id 3.
  ;; 'get-library-id 2 1' then asks for the library-id of the first ('1') CB type ('2'). The result in the example is '3'.

  report (item type-id g_sorted-first-library-index-per-type-list) + indiv-id
end

to-report calculate-cost-of-product-disposal [ product-id amount ]  ;; generic procedure
  ;; this procedure determines the revenue of selling a product with the specified product-id, with specified remaining amount, to the World Market,
  ;; or if not possible, the costs of dumping it in the Physical Environment. Costs will be reported as a positive value, revenues as a negative value
  ;; 'amount' is specified in relative (0-1) remaining use time if the product is a Tool, or quantity if it is a Consumable

  let disposal-cost 0

  ifelse can-be-sold-to-World-Market? product-id
  [ set disposal-cost (- world-market-wanted-price product-id amount) ]  ;; the price on the world market depends on the specific product and its
remaining amount
  [ if ([breed] of library-object-with-id product-id) = Tools-lib [ set amount 1 ]
    set disposal-cost (physical-environment-dump-cost product-id) * amount ]  ;; the costs of dumping in the PE depends only on the product type
  report disposal-cost
end

to-report calculate-revenue-from-selling-obsolete-stocks [ stock ] ; PB/CB/C procedure
  ;; calculates how much money the agent can expect from selling obsolete stocks (due to a possible switch she considers)

  let revenue 0
  if any? stock
  [ let product-id ([a_library-id] of one-of stock)
    let amount sum [remaining-amount] of stock


    if can-be-sold-to-World-Market? product-id
    [ set revenue world-market-wanted-price product-id amount ]
  ]
  report revenue
end

to-report replace-dashes [ word-with-dashes ]  ;; generic procedure
  ;; all that this procedure does is replace the dashes in a word by spaces, to enable communication between NetLogo names and e.g. the input text file
  ;; example: "word-with-dashes" becomes "word with dashes"

  while [ member? "-" word-with-dashes ]
  [ let dash-index position "-" word-with-dashes
    set word-with-dashes replace-item dash-index word-with-dashes " "
```

```
  ]
  report word-with-dashes
end

to-report has-product-SM?  ;; PB procedure
  ;; reports whether the agent has any actual SM defined for its a_product-based-SM

  report is-turtle? a_product-based-SM
end

to-report has-service-SM?  ;; PB procedure
  ;; reports whether the agent has any actual SM defined for its a_service-based-SM

  report is-turtle? a_service-based-SM
end

to-report concatenated-offer-price-list  ;; SM procedure
  ;; ancillary procedure to report the three elements of an offer price as a list

  report (list a_offer-price-one-time-contract-cost a_offer-price-per-unit a_offer-price-per-btu)
end

to-report is-a-Tool? [ id ]  ;; generic procedure
  ;; reports whether the calling object is a Tool(-lib)

  report (([breed] of library-object-with-id id) = Tools) or (([breed] of library-object-with-id id) = Tools-lib)
end

to-report stock-size [ stock ] ;; generic procedure
  ;; returns the size of the specified stock, measured in number of Tools or quantity of a Consumable

  let stocksize 0
  if any? stock
  [ let a-product one-of stock
    ifelse [breed] of a-product = Tools
    [ set stocksize count stock ]
    [ set stocksize [a_quantity] of a-product ]
  ]
  report stocksize
end

to-report World-Market-offered-price [ product-id ] ;; generic procedure
  ;; reports the price of buying the specified product type from the World Market

  report table:get World-Market-offered-prices-table product-id
end

to-report World-Market-wanted-price [ product-id amount ] ;; generic procedure
  ;; reports the price offered buy the World Market to buy the specified product type
  ;; 'amount' specifies the relative remaining amount (e.g. 0.5 if the Tool is 50% used)

  report (table:get World-Market-wanted-prices-table product-id) * amount
end

to-report Physical-Environment-dump-cost [ product-id ] ;; generic procedure
  ;; reports the costs of dumping the specified product type in the Physical Environment

  report table:get Physical-Environment-dump-prices-table product-id
end

to display-library-ids-and-names  ;; observer procedure
  ;; displays a list of the library id's and the name of all agents and objects in the model

  foreach g_all-library-objects-sorted-list [ if (is-agent? ?) [ ask ? [ show (word a_library-id " " a_name) ] ] ]
end

to-report most-used-associated-owned-product  ;; CM procedure
  ;; reports the owned product with the lowest remaining use time. Only works if the main input of the CM is a Tool

  report min-one-of a_associated-owned-products [ a_remaining-use-time ]
end

to-report current-tick  ;; generic procedure
  ;; reports the current tick, even if the tick counter has not started yet (in that case, report '0')

  let ctick 0
  carefully [ set ctick ticks ] [ ]
  report ctick
end

to-report World-Market-offered-prices-table  ;; generic procedure
  ;; functions as a shortcut to the table with WM offered prices

  report [a_offered-product-prices-table] of g_World-Market
end

to-report World-Market-wanted-prices-table  ;; generic procedure
  ;; functions as a shortcut to the table with WM wanted prices

  report [a_wanted-product-prices-table] of g_World-Market
end

to-report Physical-Environment-dump-prices-table  ;; generic procedure
  ;; functions as a shortcut to the table with PE dump costs

  report [a_dump-prices-table] of g_Physical-Environment
end

to-report can-be-sold-to-World-Market? [ product-id ]  ;; generic procedure
```

```
  ;; reports whether the specified product type is an element of the table with WM wanted-prices. If not, the WM cannot buy it

  report table:has-key? World-Market-wanted-prices-table product-id
end

to-report add-to-turtle-set [ element-to-add original-set ]  ;; generic procedure
  ;; just an easy-to-read helper procedure to add the specified element to the specified turtle-set and report the result

  report (turtle-set original-set element-to-add)
end

to-report remove-from-turtle-set [ element-to-remove original-set ]  ;; generic procedure
  ;; just an easy-to-read helper procedure to remove the specified element from the specified turtle-set and report the result

  report original-set with [ not (self = element-to-remove) ]
end

to-report library-object-with-id [ library-id ]  ;; generic procedure
  ;; provides a quick shortcut to the library object with the specified library id

  report item library-id g_all-library-objects-sorted-list
end

to-report Skill-object-with-id [ skill-id ]  ;; generic procedure
  ;; provides a quick shortcut to the Skill object with the specified Skill id

  report library-object-with-id get-library-id 13 skill-id
end

to-report Infrastructure-Provider-with-id [ infrastructure-id ]  ;; generic procedure
  ;; provides a quick shortcut to the Infrastructure object with the specified Infrastructure id

  report library-object-with-id get-library-id 4 infrastructure-id
end

to-report random-uniform-integer [ minimum maximum ]  ;; generic procedure
  ;; returns a random value based on a uniform distribution between 'minimum' and 'maximum'

  report minimum + random (maximum - minimum + 1)
end

to-report list-indices [ some-list ]  ;; generic procedure
  ;; returns the list indices of the specified list (e.g. [0 1 2] for a list of size 3). Useful for e.g. pairwise comparisons between two lists of
unknown size

  report n-values length some-list [?]
end

to move-SPREE  ;; fun procedure, no need to understand :)
  every 0.002
  [ ask SPREEs
    [ no-display
      set heading hheading
      fd 0.01
      set spin spin + 0.1
      set heading spin     display
      if ycor <= 57 or ycor >= 63
      [ ifelse hheading <= 180
        [ set hheading 180 - hheading ]
        [ set hheading 540 - hheading ]
      ]
      if xcor <= 19 or xcor >= 76
      [ set hheading 360 - hheading ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;                                                          ;;;;;
;;;;;                     Plots                                ;;;;;
;;;;;                                                          ;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

to plot-graphs ; observer procedure
  ;; updates all graphs in the model dashboard, based on the current market situation and agent properties

  let total-Need-in-model sum [ a_Need-per-btu] of g_all-consuming-agents

  let active-PBs-list sort-on [who] g_active-producers
  let PB-id 1

  set-current-plot "PB choices"
  foreach n-values 6 [?]
  [ set-current-plot-pen (word "sep" (? + 1))
    plot 1.5 + ?
  ]

  foreach active-PBs-list
  [ ask ?
    [ let PB-indicator (word "PB" PB-ID "-")
      let SMs (list a_product-based-SM a_service-based-SM)
      let product-based? true

      set-current-plot "PB choices"
      set-current-plot-pen (word "PB" PB-ID)

      let SM-P a_product-based-SM
```

```
      let SM-S a_service-based-SM
      let plot-id 0

      ifelse is-agent? SM-P
      [ ifelse is-agent? SM-S
        [ set plot-id [a_library-id] of SM-S - 40
          if plot-id > 2 [ set plot-id plot-id + 1 ]
        ]
        [ set plot-id [a_library-id] of SM-P - 40
          if plot-id > 1 [ set plot-id plot-id + 1 ]
          if plot-id > 4 [ set plot-id plot-id + 1 ]
        ]
      ]
      [ set plot-id [a_library-id] of SM-S - 40 + 1
        if plot-id > 4 [ set plot-id plot-id + 1 ]
      ]

      plot plot-id + a_plot-mutation

      foreach SMs
      [ let SM ?
        let pen-indicator (word PB-indicator ifelse-value product-based? ["P"] ["S"])

        set-current-plot "Market shares (relative to total Need)"
        set-current-plot-pen pen-indicator

        ifelse is-agent? SM
        [ ask SM
          [ plot (sum [ a_Need-per-btu] of g_all-consuming-agents with [ a_current-suppliers-SM = myself ]) / total-Need-in-model
            plot-pen-down
          ]
        ]
        [ plot-pen-up
          plot 0
        ]

        ifelse product-based?
        [ set-current-plot "Prices of Tools and Consumables" ]
        [ set-current-plot "Prices of Services" ]

        set-current-plot-pen pen-indicator

        ifelse is-agent? SM
        [ ask SM
          [ plot a_offer-price-per-unit
            plot-pen-down
          ]
        ]
        [ plot-pen-up
          plot 0
        ]

        set product-based? false
      ]
    ]

    set PB-id PB-id + 1
  ]

  let SMs-list sort [a_library-id] of Sales-Models-lib with [a_available?]

  let SM-id 0
  foreach SMs-list
  [ let library-id ?
    let SMs Sales-Models with [a_library-id = library-id]
    let library-object library-object-with-id library-id
    let name-of-output [a_name] of library-object-with-id ([a_main-output-id] of library-object)

    set-current-plot "Market shares"
    set-current-plot-pen name-of-output
    plot [a_market-share] of one-of g_all-sellable-products-and-services-lib with [ a_library-id = [a_main-output-id] of library-object ]

    set-current-plot ifelse-value ([a_main-output-is-Service?] of library-object) ["Average prices of Services"] ["Average prices of Tools and
Consumables"]
    set-current-plot-pen name-of-output
    ifelse any? SMs
    [ plot mean [a_offer-price-per-unit] of SMs
      plot-pen-down
    ]
    [ plot-pen-up
      plot 0
    ]
    set SM-id SM-id + 1
  ]

  set-current-plot "number of turtles in the model"
  plot count turtles

  set-current-plot "% Servicised"
  plot (sum [a_Need-per-btu] of g_all-consuming-agents with [ [a_main-input-is-Service?] of a_Consumption-Model]) / total-Need-in-model * 100

  set-current-plot "Total profit of PBs"

  set PB-id 1
  foreach active-PBs-list
  [ ask ?
    [ set-current-plot-pen (word "PB" PB-id)
      plot a_cash-balance
      set PB-id PB-id + 1
```

```
    ]
  ]
  set-current-plot-pen "x-axis"
  plot 0

  set-current-plot "Average lifestyle fit score"
  plot mean [a_current-offer-fit] of g_all-consuming-agents

  set-current-plot "Average money spent per time step for consumption"
  plot mean [a_current-offer-total-costs-per-btu] of g_all-consuming-agents ;; Or [a_total-expenditure], to plot total expenditure

  set-current-plot "Time spent per time step for consumption"
  plot sum [a_time-spent] of g_all-consuming-agents ;; Or [a_total-time-spent], to plot the total time spent on consumption

  foreach n-values g_number-of-resources [? + 1]
  [ let indiv-id ?
    let library-object library-object-with-id get-library-id 0 indiv-id

    set-current-plot "Use of resources per time step"
    set-current-plot-pen [a_name] of library-object
    plot [a_total-amount-used-as-input-this-time-step] of library-object

    set-current-plot "Associated wastes per time step"
    set-current-plot-pen [a_name] of library-object
    plot [a_total-amount-caused-as-waste-this-time-step] of library-object
  ]
end
```

F | Fictive case study: Dutch Cycling

Abbreviations / meta IDs
1 PB
2 CB
3 C
4 IP
5 WM
6 PH
7 TP
8 CP
9 S
10 MM
11 SM
12 CM
13 SK
14 PP
15 PI
16 PE
17 MD
18 MDE
19 I
20 R

PB group list
---
1 Big
2 Small

CB group list
---
1 Bicycle freight transport company

Consumer group list
---
1 Cheap and quick
2 Green and trendy
3 Convenient

Infrastructure Provider list
---
1 Electricity infrastructure provider
2 ICT infrastructure provider

Tool list
---
1 Bicycle A
2 Bicycle A2
3 Bicycle B
4 Bicycle C
5 Wheel B
6 Frame B
7 Seat B

Consumable list
---
1 Aluminium A
2 Steel A
3 Plastic A
4 Rubber A
5 Plastic C
6 CO2

Service list
---
1 Bicycle A renting
2 Bicycle B renting

Manufacturing Model list
---
1 Bicycle A production
2 Bicycle A2 production
3 Bicycle B production
4 Bicycle C (no production)
5 Buy steel

Sales Model list
---
1 Bicycle A selling
2 Bicycle A renting out
3 Bicycle B selling
4 Bicycle B renting out
5 Bicycle A2 selling
6 Bicycle C selling
7 Steel selling

Consumption Model list
---
1 Own Bicycle A
2 Rent Bicycle A
3 Own Bicycle B
4 Rent Bicycle B
5 Own Bicycle A2
6 Own Bicycle C
7 Steel-eating running (just to test)

Skill list
---
1 Bicycle production Skill
2 Bicycle assembly Skill
3 Cycling Skill
4 ICT use Skill

Policy Package list
---
1 Policy Package 1
2 Policy Package 2

Policy Instrument list
---
1 Subsidy of setting up renting scheme
2 Ban on use of Plastic A
3 Investment in ICT infrastructure
4 Environmental awareness campaign

Policy Effect list
---
1 Policy Effect 1
2 Policy Effect 2
3 Policy Effect 3
4 Policy Effect 4
5 Policy Effect 5

Market Development list
---
1 Market Development 1
2 Market Development 2

Market Development Effect list
---
1 Rubber A price increase
2 Rubber A dump costs decrease
3 Bicycle C innovation
4 'Bicycle C selling' SM introduction

Infrastructure list
---
1 Electricity access (at all desired renting spots)
2 Internet access (at all desired renting spots)

Resource list
---
1 aluminium
2 steel
3 polystyrene
4 rubber
5 bamboo
6 paint
7 CH4
8 CO2
9 heated water
10 polypropylene

ELEMENTARY UNITS

Monetary unit
"euro"

Basic time unit
'week, weeks'

Need measuring unit
'hour of transportation, hours of transportation'

Use time measuring unit
'hour, hours'

Resource

Resource ID
1

Name
aluminium

Measuring unit
kg

Resource ID
2

Name
steel

Measuring unit
kg

Resource ID
3

Name
polystyrene

Measuring unit
kg

Resource ID
4

Name
rubber

Measuring unit
kg

Resource ID
5

Name
bamboo

Measuring unit
kg

Resource ID
6

Name
NO2

Measuring unit
kg

Resource ID
7

Name
CH4

Measuring unit
gram

Resource ID
8

Name
CO2

Measuring unit
gram

Resource ID
9

Name
heated water

Measuring unit
cubic metre

Resource ID
10

Name
polypropylene

Measuring unit
kg

Producing Business

PB Group ID
1

Name
Big

Initial Models and number of agents [Manufacturing Model ID, capacity, Product-based SM ID, Product-based SM initial price (specify as [0 price-per unit 0]), Service-based SM ID, Service-based SM initial price, number of agents]
[[1 10 1 [0 320 0] 2 [0 2.7 0] 1] [3 10 3 [0 430 0] 4 [0 3.4 0] 1]]

Initial percentage of PBs that could be connected to each of the relevant infrastructures [Infrastructure ID, fraction]
[[1 1] [2 1]]

Initial percentage of PBs with infrastructure availability that are actually already connected [Infrastructure ID, fraction]
[[1 0.5] [2 1]]

Initial Skill IDs [Skill ID, fraction]
[[1 1] [4 1]]

Fraction of willingness to sacrifice expected profit for 1 point higher strategic fit
0.2

Weights for required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility, and relative (compared to market leader) market penetration [minimum, maximum]
[[0 0] [0 2] [2 4] [3 5] [0 2] [0 2] [0 0] [1 3]]

minimum thresholds for scores for required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility, and relative (compared to market leader) market penetration [minimum, maximum]
[[0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [0 0]]

Capacity over expected sales ratio
1.2

Fraction of potential consuming agents questioned during market research
0.4

Risk factor considered in market research (percentage discounted on est. demand curve) [fraction]
0.2

Return-on-investment period (BTU)
520

Period after which PB reconsiders a strategic change (BTU)
52

Period after which PB reconsiders selling price of Product/Service (BTU)
13

PB Group ID
2

Name
Small

Initial Models and number of agents [Manufacturing Model ID, capacity, Product-based SM ID, Product-based SM initial price (specify as [0 price-per unit 0]), Service-based SM ID, Service-based SM initial price, number of agents]
[[1 5 1 [0 320 0] -1 [0 0 0] 1] [2 2 5 [0 330 0] -1 [0 0 0] 1] [3 5 3 [0 430 0] -1 [0 0 0] 0] [5 50 7 [0 3 0] -1 [0 0 0] 0] [1 2 -1 [0 0 0] 2 [0 2.7 0] 1]]

Initial percentage of PBs that could be connected to each of the relevant infrastructures [Infrastructure ID, fraction]
[[1 1] [2 1]]

Initial percentage of PBs with infrastructure availability that are actually already connected [Infrastructure ID, fraction]
[[1 0.5] [2 1]]

Initial Skill IDs [Skill ID, fraction]
[[2 1]]

Fraction of willingness to sacrifice expected profit for 1 point higher strategic fit
0.2

Weights for required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility, and relative (compared to market leader) market penetration [minimum, maximum]
[[0 0] [2 5] [2 5] [1 3] [2 4] [2 5] [0 2] [0 0]]

minimum thresholds for scores for required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility, and relative (compared to market leader) market penetration [minimum, maximum]
[[0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [0 0]]

Capacity over expected sales ratio
1.5

Fraction of potential consuming agents questioned during market research
0.4

Risk factor considered in market research (percentage discounted on est. demand curve) [fraction]
0.2

Return-on-investment period (BTU)
520

Period after which PB reconsiders a strategic change (BTU)
26

Period after which PB reconsiders selling price of Product/Service (BTU)
6

Consuming Business

CB Group ID
1

Name
Bicycle freight transport company

Initial number of CBs in Group
1

Initial percentage of CBs that could be connected to each of the relevant infrastructures [Infrastructure ID, fraction]
[[1 1] [2 1]]

Initial percentage of CBs with infrastructure availability that are actually already connected [Infrastructure ID, fraction]
[[1 1] [2 1]]

Initial Skill IDs [Skill ID, fraction]
[[3 1] [4 1]]

Need (f.u./BTU) [minimum, maximum]
[1 1]

Willingness to pay for keeping with the same supplier (loyalty), relative to the overall price per btu (i.e. how much lower (as a fraction) should the price of another supplier's offer be before I will switch?)
0.05

Willingness to pay for 1 point higher strategic fit, relative to the total profit (should be lower than 0.1 otherwise you're prepared to make a loss)
0.05

Weights for required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility, and relative (compared to market leader) market penetration [minimum, maximum]
[[4 5] [0 1] [1 2] [4 5] [0 1] [0 1] [3 4] [2 5]]

minimum thresholds for scores for required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility, and relative (compared to market leader) market penetration [minimum, maximum]
[[0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [0 0]]

Default revenue value (euro/'Need measurement unit')
15

Default cost value (euro/'Need measurement unit')
10

Return-on-investment period (BTU)
250

Period after which CB reconsiders a strategic change (BTU)
75

Period after which CB reconsiders supplier (BTU)
52

Consumer

Consumer Group ID
1

Name
Cheap and quick

Initial number of Consumers in Group
50

Initial percentage of Consumers that could be connected to each of the relevant infrastructures [Infrastructure ID, fraction]
[[1 1] [2 1]]

Initial percentage of Cs with infrastructure availability that are actually already connected [Infrastructure ID, fraction]
[[1 1] [2 0.95]]

Initial Skill IDs [Skill ID, fraction]
[[3 1] [4 1]]

Need (f.u./BTU) [minimum, maximum]
[1 2]

Maximum threshold for total costs per functional unit of need (euro/functional unit of need)
5

Willingness to pay for keeping with the same supplier (loyalty), relative to the overall price per btu (i.e. how much lower (as a fraction) should the price of another supplier's offer be before I will switch?)
0.05

Willingness to pay for 1 point higher lifestyle fit, relative to the total consumption cost per btu
0

Weights for required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility, and relative (compared to market leader) market penetration
[[4 5] [0 1] [1 2] [4 5] [0 1] [0 1] [3 4] [0 1]]

minimum thresholds for scores for required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility, and relative (compared to market leader) market penetration
[[0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [0 0]]

Return-on-investment period (BTU)
520

Period after which Consumer reconsiders a strategic change (BTU)
104

Period after which Consumer reconsiders supplier (BTU)
52

Consumer Group ID
2

Name
Green and trendy

Initial number of Consumers in Group
25

Initial percentage of Consumers that could be connected to each of the relevant infrastructures [Infrastructure ID, fraction]
[[1 1] [2 1]]

Initial percentage of Cs with infrastructure availability that are actually already connected [Infrastructure ID, fraction]
[[1 0.8] [2 1]]

Initial Skill IDs [Skill ID, fraction]
[[3 1] [4 1]]

Need (f.u./BTU) [minimum, maximum]
[1 2]

Maximum threshold for total costs per functional unit of need (euro/functional unit of need)
5

Willingness to pay for keeping with the same supplier (loyalty), relative to the overall price per btu (i.e. how much lower (as a fraction) should the price of another supplier's offer be before I will switch?)
0.05

Willingness to pay for 1 point higher lifestyle fit, relative to the total consumption cost per btu
0.4

Weights for required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility, and relative (compared to market leader) market penetration
[[0 0] [3 5] [0 1] [1 2] [1 5] [3 5] [0 0] [0 1]]

minimum thresholds for scores for required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility, and relative (compared to market leader) market penetration
[[0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [0 0]]

Return-on-investment period (BTU)
520

Period after which Consumer reconsiders a strategic change (BTU)
104

Period after which Consumer reconsiders supplier (BTU)
52

Consumer Group ID
3

Name
Convenient

Initial number of Consumers in Group
50

Initial percentage of Consumers that could be connected to each of the relevant infrastructures [Infrastructure ID, fraction]
[[1 1] [2 1]]

Initial percentage of Cs with infrastructure availability that are actually already connected [Infrastructure ID, fraction]
[[1 0.9] [2 0.7]]

Initial Skill IDs [Skill ID, fraction]
[[3 1]]

Need (f.u./BTU) [minimum, maximum]
[1 2]

Maximum threshold for total costs per functional unit of need (euro/functional unit of need)
5

Willingness to pay for keeping with the same supplier (loyalty), relative to the overall price per btu (i.e. how much lower (as a fraction) should the price of another supplier's offer be before I will switch?)
0.1

Willingness to pay for 1 point higher lifestyle fit, relative to the total consumption cost per btu
0.2

Weights for required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility, and relative (compared to market leader) market penetration
[[0 0] [1 2] [4 5] [2 3] [0 0] [0 0] [0 0] [0 2]]

minimum thresholds for scores for required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility, and relative (compared to market leader) market penetration
[[0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [0 0] [0 0]]

Return-on-investment period (BTU)
520

Period after which Consumer reconsiders a strategic change (BTU)
208

Period after which Consumer reconsiders supplier (BTU)
104

Infrastructure Provider

Infrastructure Provider ID
1

Name
Electricity infrastructure provider

Operated infrastructure [Infrastructure ID]
1

One-time connection fee (euro)
0

Infrastructure Provider ID

2

Name
ICT infrastructure provider

Operated infrastructure [Infrastructure ID]
2

One-time connection fee (euro)
50

World Market

List of Products that can be bought from the World Market [Type, ID, price]
[[8 1 2] [8 2 0.6] [8 3 1.2] [8 4 1] [8 5 3] [7 1 400] [7 4 430] [7 5 40] [7 6 100] [7 7 25]]

List of Products that can be sold to the World Market [Type, ID, price]
[[8 1 1] [8 2 0.3] [7 1 200] [7 2 250] [7 3 150] [7 4 250] [7 5 20] [7 6 100] [7 7 12.5]]

Physical Environment

List of Products that are dumped at end-of-life [Type, ID, dump costs]
[[8 3 0.1] [8 4 0.1] [8 5 0.1] [8 6 0]]

Tool

Tool ID
1

Name
Bicycle A

Initially available?
1

Use time (BTU)
208

Use of resources [Resource ID, number]
[ ]

Associated wastes [Resource ID, number]
[ ]

Scores on required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility
[8 3 4 10 3 3 2]

Initial market penetration (what is the product's real-world market share? This can function as a reality check of initial consumer choices!
0.4

Tool ID
2

Name
Bicycle A2

Initially available?
1

Use time (BTU)
208

Use of resources [Resource ID, number]
[ ]

Associated wastes [Resource ID, number]
[ ]

Scores on required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility
[8 6 4 10 3 3 2]

Initial market penetration (what is the product's real-world market share? This can function as a reality check of initial consumer choices!
0.1

Tool ID
3

Name
Bicycle B

Initially available?
1

Use time (BTU)
208

Use of resources [Resource ID, number]
[ ]

Associated wastes [Resource ID, number]
[ ]

Scores on required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility
[8 3 7 10 6 5 2]

Initial market penetration (what is the product's real-world market share? This can function as a reality check of initial consumer choices!
0.4

Tool ID
4

Name
Bicycle C

Initially available?
0

Use time (BTU)
150

Use of resources [Resource ID, number]
[[4 6] [5 2]]

Associated wastes [Resource ID, number]
[[8 4.5] [9 4.5]]

Scores on required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility
[8 8 6 10 9 9 2]

Initial market penetration (what is the product's real-world market share? This can function as a reality check of initial consumer choices!
0

Tool ID
5

Name
Wheel B

Initially available?
1

Use time (BTU)
500

Use of resources [Resource ID, number]
[[1 0.5] [2 0.5] [4 0.2]]

Associated wastes [Resource ID, number]

[[7 5] [8 2.5] [9 2.5]]

Scores on required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility
[ ]

Initial market penetration (what is the product's real-world market share? This can function as a reality check of initial consumer choices!
0

Tool ID
6

Name
Frame B

Initially available?
1

Use time (BTU)
500

Use of resources [Resource ID, number]
[[1 6.5] [2 4.5] [3 1.5]]

Associated wastes [Resource ID, number]
[[7 5] [8 5] [9 5]]

Scores on required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility
[ ]

Initial market penetration (what is the product's real-world market share? This can function as a reality check of initial consumer choices!
0

Tool ID
7

Name
Seat B

Initially available?
1

Use time (BTU)
300

Use of resources [Resource ID, number]
[[3 0.5] [4 0.1]]

Associated wastes [Resource ID, number]
[[8 0.5] [9 0.1]]

Scores on required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility
[ ]

Initial market penetration (what is the product's real-world market share? This can function as a reality check of initial consumer choices!
0

Consumable

Consumable ID
1

Name
Aluminium A

Initially available?
1

Measuring unit
kg

Use of resources [Resource ID, number]
[[1 1]]

Associated wastes [Resource ID, number]
[[7 0.5] [8 1.5] [9 0.75]]

Scores on required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility
[ ]

Initial market penetration (what is the product's real-world market share? This can function as a reality check of initial consumer choices!
0

Consumable ID
2

Name
Steel A

Initially available?
1

Measuring unit
kg

Use of resources [Resource ID, number]
[[2 1]]

Associated wastes [Resource ID, number]
[[7 0.5] [8 1.5] [9 0.75]]

Scores on required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility
[6 2 6 6 1 1 4]

Initial market penetration (what is the product's real-world market share? This can function as a reality check of initial consumer choices!
0.05

Consumable ID
3

Name
Plastic A

Initially available?
1

Measuring unit
kg

Use of resources [Resource ID, number]
[[3 1]]

Associated wastes [Resource ID, number]
[[7 0.2] [8 0.5] [9 0.5]]

Scores on required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility
[ ]

Initial market penetration (what is the product's real-world market share? This can function as a reality check of initial consumer choices!
0

Consumable ID
4

Name
Rubber A

Initially available?
1

Measuring unit
kg

Use of resources [Resource ID, number]
[[4 1]]

Associated wastes [Resource ID, number]
[[7 0.1] [8 0.1] [9 0.1]]

Scores on required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility
[ ]

Initial market penetration (what is the product's real-world market share? This can function as a reality check of initial consumer choices!
0

Consumable ID
5

Name
Plastic C

Initially available?
1

Measuring unit
kg

Use of resources [Resource ID, number]
[[10 1]]

Associated wastes [Resource ID, number]
[[8 0.1] [9 0.1]]

Scores on required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility
[ ]

Initial market penetration (what is the product's real-world market share? This can function as a reality check of initial consumer choices!
0

Consumable ID
6

Name
CO2

Initially available?
1

Measuring unit
g

Use of resources [Resource ID, number]
[[8 1]]

Associated wastes [Resource ID, number]
[ ]

Scores on required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility
[ ]

Initial market penetration (what is the product's real-world market share? This can function as a reality check of initial consumer choices!
0

Service

Service ID
1

Name
Bicycle A sharing

Initially available?
1

Functional unit
Hour of one rented bicycle

Use of resources [Resource ID, number]
[ ]

Associated wastes [Resource ID, number]
[ ]

Scores on required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility
[5 8 7 5 6 6 8]

Initial market penetration (what is the service's real-world market share? This can function as a reality check of initial consumer choices!
0

Service ID
2

Name
Bicycle B sharing

Initially available?
1

Functional unit
Hour of one rented bicycle

Use of resources [Resource ID, number]
[ ]

Associated wastes [Resource ID, number]
[ ]

Scores on required time, environmental profile, user-friendliness & comfort, availability, status, innovativeness, flexibility
[5 8 8 5 7 7 8]

Initial market penetration (what is the service's real-world market share? This can function as a reality check of initial consumer choices!
0.1

Manufacturing Model

Manufacturing Model ID
1

Name
Bicycle A production

Initially available?
1

Main output [Type, ID]
[7 1]

Secondary inputs [Type, ID, ratio]
[[8 1 7.5] [8 2 5.5] [8 3 2] [8 4 0.5]]

Secondary outputs [Type, ID, ratio]
[[8 6 0.1]]

Associated product-based Sales Model id
1

Possible service-based Sales Model ids [Sales Model ID]
[2]

Structural costs (euro/capacity unit/BTU)
1

One-time investment costs (euro)
2000

Labour time (BTU/unit)
0.5

Other costs based on output (euro/unit)
200

Required Skills [Skill ID]
[1]

Required infrastructures [Infrastructure ID]
[ ]

Manufacturing Model ID
2

Name
Bicycle A2 production

Initially available?
1

Main output [Type, ID]
[7 2]

Secondary inputs [Type, ID, ratio]
[[8 1 7.5] [8 2 5.5] [8 4 0.5] [8 5 2]]

Secondary outputs [Type, ID, ratio]
[[8 6 0.1]]

Associated product-based Sales Model id
5

Possible service-based Sales Model ids [Sales Model ID]
[ ]

Structural costs (euro/capacity unit/BTU)
1

One-time investment costs (euro)
2000

Labour time (BTU/unit)
0.5

Other costs based on output (euro/unit)
210

Required Skills [Skill ID]
[1]

Required infrastructures [Infrastructure ID]
[ ]

Manufacturing Model ID
3

Name
Bicycle B production

Initially available?
1

Main output [Type, ID]
[7 3]

Secondary inputs [Type, ID, ratio]
[[7 5 2] [7 6 1] [7 7 1]]

Secondary outputs [Type, ID, ratio]
[[8 6 0.1]]

Associated product-based Sales Model id
3

Possible service-based Sales Model ids [Sales Model ID]
[4]

Structural costs (euro/capacity unit/BTU)
1

One-time investment costs (euro)
2000

Labour time (BTU/unit)
0.5

Other costs based on output (euro/unit)
100

Required Skills [Skill ID]
[1]

Required infrastructures [Infrastructure ID]
[ ]

Manufacturing Model ID
4

Name
Bicycle C (no production)

Initially available?
0

Main output [Type, ID]
[7 4]

Secondary inputs [Type, ID, ratio]
[[7 4 1]]

Secondary outputs [Type, ID, ratio]
[ ]

Associated product-based Sales Model id
6

Possible service-based Sales Model ids [Sales Model ID]
[ ]

Structural costs (euro/capacity unit/BTU)
0.1

One-time investment costs (euro)
2000

Labour time (BTU/unit)
0

Other costs based on output (euro/unit)
0

Required Skills [Skill ID]
[ ]

Required infrastructures [Infrastructure ID]
[ ]

Manufacturing Model ID
5

Name
Buy steel

Initially available?
0

Main output [Type, ID]
[8 2]

Secondary inputs [Type, ID, ratio]
[[8 2 1]]

Secondary outputs [Type, ID, ratio]
[ ]

Associated product-based Sales Model id
7

Possible service-based Sales Model ids [Sales Model ID]
[ ]

Structural costs (euro/capacity unit/BTU)
0.1

One-time investment costs (euro)
2000

Labour time (BTU/unit)
0

Other costs based on output (euro/unit)
2

Required Skills [Skill ID]
[ ]

Required infrastructures [Infrastructure ID]
[ ]

Sales Model

Sales Model ID
1

Name
Bicycle A selling

Initially available?
1

Corresponding Manufacturing Model [Manufacturing Model ID]
1

Main output [Type, ID]
[7 1]

Main input/stock type [Type, ID, ratio to main output]
[7 1 1]

Secondary inputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Secondary outputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Recoverable wastes per main input Product [Type, ID, ratio]
[ ]

Stock / demand ratio [fraction]
-1

Use time multiplier for main input Product (only different than 1 if service-based!)
1

Structural costs (euro/BTU)
0.1

One-time investment costs (euro)
50

Labour time (BTU/measuring unit)
0.25

Other costs based on output (euro/measuring unit)
25

Required Skills [Skill ID]
[ ]

Required infrastructures [Infrastructure ID]
[ ]

Sales Model ID
2

Name
Bicycle A renting out

Initially available?
1

Corresponding Manufacturing Model [Manufacturing Model ID]
1

Main output [Type, ID]
[9 1]

Main input/stock type [Type, ID, ratio to main output]
[7 1 1]

Secondary inputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Secondary outputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Recoverable wastes per main input Product [Type, ID, ratio]
[[8 1 7] [8 2 5]]

Stock / demand ratio [fraction]
0.1

Use time multiplier for main input Product (only different than 1 if service-based!)
1.2

Structural costs (euro/BTU)
0.1

One-time investment costs (euro)
50

Labour time (BTU/measuring unit)
0.5

Other costs based on output (euro/measuring unit)
1.6

Required Skills [Skill ID]
[4]

Required infrastructures [Infrastructure ID]
[1 2]

Sales Model ID
3

Name
Bicycle B selling

Initially available?
1

Corresponding Manufacturing Model [Manufacturing Model ID]
3

Main output [Type, ID]

[7 3]

Main input/stock type [Type, ID, ratio to main output]
[7 3 1]

Secondary inputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Secondary outputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Recoverable wastes per main input Product [Type, ID, ratio]
[ ]

Stock / demand ratio [fraction]
-1

Use time multiplier for main input Product (only different than 1 if service-based!)
1

Structural costs (euro/BTU)
0.1

One-time investment costs (euro)
50

Labour time (BTU/measuring unit)
0.25

Other costs based on output (euro/measuring unit)
25

Required Skills [Skill ID]
[ ]

Required infrastructures [Infrastructure ID]
[ ]

Sales Model ID
4

Name
Bicycle B renting out

Initially available?
1

Corresponding Manufacturing Model [Manufacturing Model ID]
3

Main output [Type, ID]
[9 2]

Main input/stock type [Type, ID, ratio to main output]
[7 3 1]

Secondary inputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Secondary outputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Recoverable wastes per main input Product [Type, ID, ratio]
[ ]

Stock / demand ratio [fraction]
0.1

Use time multiplier for main input Product (only different than 1 if service-based!)
1.2

Structural costs (euro/BTU)
0.1

One-time investment costs (euro)
50

Labour time (BTU/measuring unit)
0.5

Other costs based on output (euro/measuring unit)
1.5

Required Skills [Skill ID]
[4]

Required infrastructures [Infrastructure ID]
[1 2]

Sales Model ID
5

Name
Bicycle A2 selling

Initially available?
1

Corresponding Manufacturing Model [Manufacturing Model ID]
2

Main output [Type, ID]
[7 2]

Main input/stock type [Type, ID, ratio to main output]
[7 2 1]

Secondary inputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Secondary outputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Recoverable wastes per main input Product [Type, ID, ratio]
[ ]

Stock / demand ratio [fraction]
-1

Use time multiplier for main input Product (only different than 1 if service-based!)
1

Structural costs (euro/BTU)
0.1

One-time investment costs (euro)
50

Labour time (BTU/measuring unit)
0.25

Other costs based on output (euro/measuring unit)
25

Required Skills [Skill ID]
[ ]

Required infrastructures [Infrastructure ID]
[ ]

Sales Model ID
6

Name
Bicycle C selling

Initially available?
0

Corresponding Manufacturing Model [Manufacturing Model ID]
4

Main output [Type, ID]
[7 4]

Main input/stock type [Type, ID, ratio to main output]
[7 4 1]

Secondary inputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Secondary outputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Recoverable wastes per main input Product [Type, ID, ratio]
[ ]

Stock / demand ratio [fraction]
-1

Use time multiplier for main input Product (only different than 1 if service-based!)
1

Structural costs (euro/BTU)
0.1

One-time investment costs (euro)
50

Labour time (BTU/measuring unit)
0.25

Other costs based on output (euro/measuring unit)
25

Required Skills [Skill ID]
[ ]

Required infrastructures [Infrastructure ID]
[ ]

Sales Model ID
7

Name
Steel selling

Initially available?
0

Corresponding Manufacturing Model [Manufacturing Model ID]
5

Main output [Type, ID]
[8 2]

Main input/stock type [Type, ID, ratio to main output]
[8 2 1]

Secondary inputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Secondary outputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Recoverable wastes per main input Product [Type, ID, ratio]
[ ]

Stock / demand ratio [fraction]
-1

Use time multiplier for main input Product (only different than 1 if service-based!)
1

Structural costs (euro/BTU)
0.1

One-time investment costs (euro)
50

Labour time (BTU/measuring unit)
0.25

Other costs based on output (euro/measuring unit)
0

Required Skills [Skill ID]
[ ]

Required infrastructures [Infrastructure ID]
[ ]

Consumption Model

Consumption Model ID
1

Name
Own Bicycle A

Initially available?
1

Automatic repurchase?
0

Main input/stock type [Type, ID, ratio to unit of Need]
[7 1 1]

Secondary inputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Secondary outputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Revenue improvement multiplier (number)
1

Structural costs (euro/BTU)
1

One-time investment costs (euro)
2.5

Labour time (BTU/measuring unit)
0.5

Other costs based on output (euro/measuring unit)
0

Required Skills [Skill ID]
[3]

Required infrastructures [Infrastructure ID]
[1]

Consumption Model ID
2

Name
Rent Bicycle A

Initially available?
1

Automatic repurchase?
1

Main input/stock type [Type, ID, ratio to unit of Need]
[9 1 1]

Secondary inputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Secondary outputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Revenue improvement multiplier (number)

1

Structural costs (euro/BTU)
0

One-time investment costs (euro)
2.5

Labour time (BTU/measuring unit)
1

Other costs based on output (euro/measuring unit)
0

Required Skills [Skill ID]
[3 4]

Required infrastructures [Infrastructure ID]
[1 2]

Consumption Model ID
3

Name
Own Bicycle B

Initially available?
1

Automatic repurchase?
0

Main input/stock type [Type, ID, ratio to unit of Need]
[7 3 1]

Secondary inputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Secondary outputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Revenue improvement multiplier (number)
1

Structural costs (euro/BTU)
1

One-time investment costs (euro)
2.5

Labour time (BTU/measuring unit)
0.5

Other costs based on output (euro/measuring unit)
0

Required Skills [Skill ID]
[3]

Required infrastructures [Infrastructure ID]
[1]

Consumption Model ID
4

Name
Rent Bicycle B

Initially available?
1

Automatic repurchase?
1

Main input/stock type [Type, ID, ratio to unit of Need]
[9 2 1]

Secondary inputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Secondary outputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Revenue improvement multiplier (number)
1

Structural costs (euro/BTU)
0

One-time investment costs (euro)
2.5

Labour time (BTU/measuring unit)
1

Other costs based on output (euro/measuring unit)
0

Required Skills [Skill ID]
[3 4]

Required infrastructures [Infrastructure ID]
[1 2]

Consumption Model ID
5

Name
Own Bicycle A2

Initially available?
1

Automatic repurchase?
0

Main input/stock type [Type, ID, ratio to unit of Need]
[7 2 1]

Secondary inputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Secondary outputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Revenue improvement multiplier (number)
1

Structural costs (euro/BTU)
1

One-time investment costs (euro)
2.5

Labour time (BTU/measuring unit)
0.5

Other costs based on output (euro/measuring unit)
0

Required Skills [Skill ID]
[3]

Required infrastructures [Infrastructure ID]
[1]

Consumption Model ID
6

Name
Own Bicycle C

Initially available?
0

Automatic repurchase?
0

Main input/stock type [Type, ID, ratio to unit of Need]
[7 4 1]

Secondary inputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Secondary outputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Revenue improvement multiplier (number)
1

Structural costs (euro/BTU)
1

One-time investment costs (euro)
2.5

Labour time (BTU/measuring unit)
0.5

Other costs based on output (euro/measuring unit)
0

Required Skills [Skill ID]
[3]

Required infrastructures [Infrastructure ID]
[1]

Consumption Model ID
7

Name
Steel-eating running (just to test)

Initially available?
0

Automatic repurchase?
1

Main input/stock type [Type, ID, ratio to unit of Need]
[8 2 0.1]

Secondary inputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Secondary outputs (measuring unit/unit) [Type, ID, ratio]
[ ]

Revenue improvement multiplier (number)
1

Structural costs (euro/BTU)
0

One-time investment costs (euro)
0

Labour time (BTU/measuring unit)
0.5

Other costs based on output (euro/measuring unit)
2.3

Required Skills [Skill ID]
[ ]

Required infrastructures [Infrastructure ID]
[ ]

Skill

Skill ID
1

Name
Bicycle production Skill

Skill learning costs (euro)
1000

Skill ID
2

Name
Bicycle assembly Skill

Skill learning costs (euro)
500

Skill ID
3

Name
Cycling Skill

Skill learning costs (euro)
10

Skill ID
4

Name
ICT use Skill

Skill learning costs (euro)
10

Policy Package

Policy Package ID
1

Name
Policy Package 1

PP activation trigger [element, compare, value]
["time tick" "is equal to" 0]

PP deactivation trigger [element, compare, value]
[ ]

List of Policy Instruments [Policy Instrument ID]
[1 3]

Policy Package ID
2

Name
Policy Package 2

PP activation trigger [element, compare, value]
["time tick" "is equal to" 0]

PP deactivation trigger [element, compare, value]
[ ]

List of Policy Instruments [Policy Instrument ID]

[2 4]

Policy Instrument

Policy Instrument ID
1

Name
Subsidy for setting up a renting scheme

Short name (label)
Renting scheme subsidy

PI activation trigger [element, compare, value]
[ ]

PI deactivation trigger [element, compare, value]
[ ]

List of Policy Effects [Policy Effect ID]
[1]

Policy Instrument ID
2

Name
Tax on use of Plastic A

Short name (label)
Plastic A tax

PI activation trigger [element, compare, value]
[ ]

PI deactivation trigger [element, compare, value]
[ ]

List of Policy Effects [Policy Effect ID]
[2]

Policy Instrument ID
3

Name
Environmental awareness campaign

Short name (label)
Env. Aware. campaign

PI activation trigger [element, compare, value]
[ ]

PI deactivation trigger [element, compare, value]
[ ]

List of Policy Effects [Policy Effect ID]
[5]

Policy Effect

Policy Effect ID
1

Name
Policy Effect 1

Temporary effect? [Boolean]
Yes

Target group [Type]
11

Criterion [attribute, compare, value]
['Corresponding Product or Service' 'is equal to' 'Bicycle A use time' OR 'Bicycle B use time']

Effect [attribute, change, value]
['One-time investments costs' 'minus' 10000]

Policy Effect ID
2

Name
Policy Effect 2

Temporary effect? [Boolean]
Yes

Target group [Type]
5

Criterion [attribute, compare, value]
[ ]

Effect [attribute, change, value]
['offered price' 'plus' 2.5]

Policy Effect ID
3

Name
Policy Effect 3

Temporary effect? [Boolean]
No

Target group [Type]
3

Criterion [attribute, compare, value]
[ ]

Effect [attribute, change, value]
['Weight for environmental profile indication' 'plus' 2]

Market Development

Market Development ID
1

Name
Market Development 1

Short name
MD1

MD Event activation trigger [element, compare, value]
['time tick' 'is equal to' 250]

List of Market Development Effects [Market Development Effect ID]
[1 2]

Market Development ID
2

Name
Market Development 2

Short name
MD2

MD Event activation trigger [element, compare, value]
['time tick' 'is equal to' 400]

List of Market Development Effects [Market Development Effect ID]

[3 4]

Market Development Effect

Market Development Effect ID
1

Name
Rubber A price increase

Target group [Type]
5

Criterion [attribute, compare, value]
[ ]

Effect [attribute, change, value]
["Selling price of Rubber A" "plus" 2.5]

Market Development Effect ID
2

Name
Rubber A dump costs decrease

Target group [Type]
6

Criterion [attribute, compare, value]
[ ]

Effect [attribute, change, value]
["Dump costs of Rubber A" "minus" 2]

Market Development Effect ID
3

Name
Bicycle C innovation

Target group [Type]
7

Criterion [attribute, compare, value]
["Tool ID" "is equal to" TP4]

Effect [attribute, change, value]
["Availability?" "becomes" TRUE]

Market Development Effect ID
4

Name
'Bicycle C selling' SM introduction

Target group [Type]
11

Criterion [attribute, compare, value]
["Service Model ID" "is equal to" SM6]

Effect [attribute, change, value]
["Availability?" "becomes" TRUE]

---
end of file