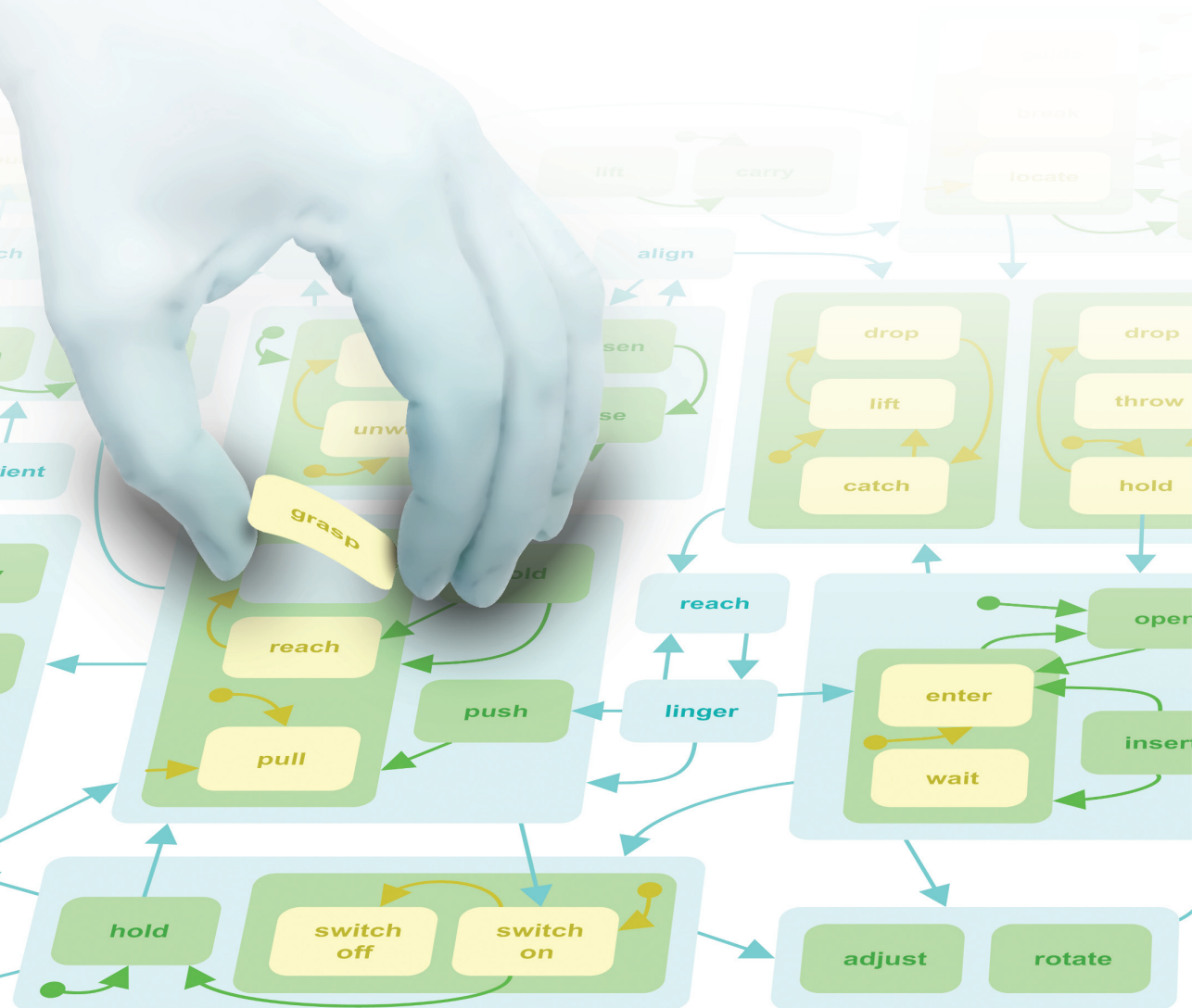


Testing virtual use with scenarios



Wilfred van der Vegte

Testing virtual use with scenarios

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. dr. ir. J.T. Fokkema,
voorzitter van het college van promoties,
op dinsdag 15 december 2009 te 10.00 uur
door

Wilhelm Frederik van der VEGTE,

werktuigkundig ingenieur,
Master of Technological Design
geboren te Leeuwarden

Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. I. Horváth.

Samenstelling promotiecommissie:

Rector Magnificus, voorzitter

Prof. dr. I. Horváth, Technische Universiteit Delft, promotor

Prof. dr. C. Spitas, Technische Universiteit Delft

Prof. dr. D. Talabă, Universitatea Transilvania Brasov

Prof. dr. K. Váradi, Budapesti Műszaki és Gazdaságtudományi Egyetem

Prof. dr. P. Vink, Technische Universiteit Delft

Prof. dr. ir. F.J.A.M. van Houten, Universiteit Twente

Prof. dr. ir. R.H.M. Goossens, Technische Universiteit Delft

Prof. dr. ir. J.P.M. Geraedts, Technische Universiteit Delft (reserveid)

Testing virtual use with scenarios

Web version. Contains movie clip: see page 193

Wilhelm F. van der Vegte

PhD thesis, Delft University of Technology, Delft, the Netherlands

ISBN: 978-90-6562-097-2

Published and distributed by VSSD, Delft, www.vssd.nl/hlf

Copyright © 2009 W.F. van der Vegte

w.f.vandervegte@tudelft.nl.

TABLE OF CONTENTS

Table of contents	i
List of Figures	iv
List of tables	vi
1 Introduction	1
1.1 Simulating the use of products: Setting the stage	1
1.2 Open issues in simulation of use processes	3
1.3 Research problems	4
1.4 Scope of this research, targeted deliverables, and criteria	5
1.5 Research questions	6
1.6 Research postulate and hypotheses	8
1.7 Research methods and structure of the thesis	9
1.8 Forerunning publications	11
2 Knowledge exploration and literature study	13
2.1 General introduction	13
2.2 Introduction to behavioural simulations	14
2.3 Review of behavioural simulations of artefacts and humans in time	20
2.4 Introduction to the review of control in simulations	36
2.5 Review of control of simulations and the application of scenarios to simulation	39
2.6 Discussion of the findings of the literature study	49
3 Concepts and theory of resource-integrated interaction simulation	55
3.1 Introduction	55
3.2 Conversion of the conceptual reasoning model of human-artefact interaction to a computational concept for simulation of interactions	56
3.3 Resource-integrated modelling and simulation	57
3.4 Nucleus-based representation of humans and artefacts in the modelling layer	59

3.5	The behaviour layer: simulating behaviour with nucleus-based models	64
3.6	The logistics layer	65
3.7	Concluding remarks	75
4	Prototype implementation	77
4.1	Objectives	77
4.2	Conversion of theoretical resources to the requested functionality	77
4.3	Selection of tools for a proof-of-concept implementation	84
4.4	Building proxies of nucleus-based simulation models with Adams	90
4.5	Specifying surrogate nucleus-based control instructions with Simulink Stateflow	97
4.6	Implementation of the signal conversion specification	103
4.7	Interfacing the control and the simulation model	105
4.8	Overview of modelling elements, specification elements and their mutual relations in the proof-of-concept prototype	105
5	Application and testing of the proof-of- concept implementation	107
5.1	Objectives	107
5.2	Approach	107
5.3	Instantiation of behavioural simulation models	109
5.4	Instantiation of control models and specifications	110
5.5	Instantiation of interfacing between control and simulation	114
5.6	Running controlled physical simulations	114
5.7	Discussion and conclusions of application testing	116
6	Justification of the supporting theory	119
7	Validation	125
7.1	Objectives	125
7.2	Extended functional affordances of scenario-bundle based control of simulations	125
7.3	Convenience of use from technical aspects	127
7.4	Conclusions concerning the validation	133
8	Conclusions and recommendations	135
8.1	Introduction	135
8.2	Conclusions	135
8.3	Recommendations for future research and development	139

Summary	143
Samenvatting	151
References	159
Index	175
Appendix 1 – Glossary	179
Appendix 2 – List of symbols	183
Appendix 3 – Processing flowchart of the human interaction construct and the procedure structure	185
Appendix 4 – Entity-relationship diagram of the proof-of- concept implementation	186
Appendix 5 – Detailed Simulink constructs of the third composite sample case (snack dispenser)	189
Appendix 6 –Movie clip	193
Acknowledgments	195
Curriculum vitae	197

List of Figures

Figure 1.	Key concepts of the approach presented in this thesis	2
Figure 2.	Representations in use for various simulation approaches	3
Figure 3.	Three-layered concept of modelling, simulation, and logistic control of simulations.	8
Figure 4.	Conceptual reasoning model of human-artefact interaction	9
Figure 5.	Nine-stage scheme of design-inclusive research	10
Figure 6.	Taxonomy of model types used in simulation	16
Figure 7.	Taxonomy of the areas of physics	18
Figure 8.	Typification of human behaviour based on elementary functions, including the relations with the involved body parts and organs.	19
Figure 9.	Output of qualitative simulation of vegetation growth with the simulation package VisiGARP	23
Figure 10.	Mass-spring-damper system with Simulink block diagram and simulation output	24
Figure 11.	Schematics of the dynamics of human sagittal balance with bond graph model	26
Figure 12.	Skeleton-like model of a hand drill	27
Figure 13.	Multibody simulation model of a test rig for motorcycles	28
Figure 14.	Multiphysics simulation of a switch.	30
Figure 15.	FE-based simulation of consumer durables.	30
Figure 16.	Combination of a mesh-based deformable skin model with a kinematical skeleton-like model in a computer game	31
Figure 17.	Discrete interpretation as output of a continuous change in voltage	37
Figure 18.	Generic stages of human control in interaction inserted into the diagram of Figure 8.	38
Figure 19.	Simulation of a human falling from a balcony, created with Endorphin	41
Figure 20.	Mapping of hypotheses to theoretical solution elements.	56
Figure 21.	Processing scheme of three-layered resource-integrated modelling and simulation	58
Figure 22.	Ontological conceptualization of a nucleus	60
Figure 23.	Example of a logical construct	68
Figure 24.	Example of transitions, situations, and states of a logical construct	72
Figure 25.	Occurrences of meter events e_1 with increasing orientation, e_2 with decreasing orientation, and e_3 with bidirectional orientation	73
Figure 26.	Realization of a delayed transition in a logical construct.	75
Figure 27.	Signal flows of controlled interaction simulation	79
Figure 28.	Decomposition of the signal conversion specification.	80
Figure 29.	Flowchart representation of controlled simulation	83
Figure 30.	Low-resolution nucleus-based model of a human finger and an object to interact with, shown as a simplified 2D representation	91
Figure 31.	Contact normal force N acting on an infinitesimal half-space HS at an offset from the reference point of a boundary particle. <i>a.</i> overview; <i>b.</i> free-body diagram	93

Figure 32.	Sphere acting as half-space to eliminate the offset shown in Figure 31. <i>a.</i> overview; <i>b.</i> free-body diagram	93
Figure 33.	Example of a Simulink block representing a physically-based simulation, with two control signals as input and five meter signals as output	97
Figure 34.	<i>a.</i> Excerpt from an example Stateflow chart with its graphical specification elements used in the proof-of-concept implementation. <i>b.</i> statechart equivalent of the Stateflow chart. <i>c</i> and <i>d.</i> alternative representations of Stateflow chart based on subcharting.	98
Figure 35.	<i>a.</i> Example of a Stateflow 'Chart' block with external connections in Matlab Simulink. <i>b.</i> specification in the Simulink model explorer	101
Figure 36.	Layered structuring of the human interaction construct (HIC)	102
Figure 37.	Example implementation of a signal conversion specification in Simulink	103
Figure 38.	Contents of the human stimulus recognition block in Figure 37	104
Figure 39.	Specification of a delay	104
Figure 40.	Simulink block diagram in which all the constructs for resource-integrated simulation have been included and connected.	106
Figure 41.	Scenario-based simulation of a basic interaction: pushing and releasing a button.	107
Figure 42.	Simulations of using two versions <i>a</i> and <i>b</i> of a pedal bin	108
Figure 43.	<i>a.</i> Simulation of throwing an object into an open garbage bin. <i>b.</i> End result of the same simulation after missing. The scenario bundle is shown at the top right.	109
Figure 44.	Concept design of a snack dispenser	110
Figure 45.	Scenario bundle of the dispenser	111
Figure 46.	Scenario layer and response selection layer of human control in using the snack dispenser.	111
Figure 47.	Response-execution layer of the HIC of the use of the snack dispenser.	112
Figure 48.	Procedure structure specifying the control mechanisms of the snack dispenser.	113
Figure 49.	Compilation of simulation frames: <i>a.</i> overview; <i>b</i> and <i>c.</i> detailed views of grasping.	114
Figure 50.	XY plots providing feedback during computation of the physical simulation.	115
Figure 51.	Simulated paths in the scenario bundle (<i>cf.</i> Figure 46)	116
Figure 52.	Identified limitations of the theory based on reasoning with its consequences.	123
Figure 53.	Minimum value for the preparation efforts indicator, PEI, as a function of the number of model variations, n , and the number of investigated transitions, N_T .	130
Figure 54.	Minimum values for the simulation time index, STI, as a function of the number of moving entities in the model, n_{mov} , and the number of investigated transitions, N_T .	132
Figure 55.	Simplified reasoning model of human-product interaction, and arrangement of models and specifications in scenario bundle-based simulation	145

List of tables

Table 1.	Comparison of behavioural simulation approaches	34
Table 2.	Review of integral human simulations based on their capabilities	48
Table 3.	Geometric meanings of the levels of modelling entities	61
Table 4.	Key relations in solid mechanics	63
Table 5.	Principal constructs of resource-integrated modelling and simulation	78
Table 6.	Specification of the labels used in Figure 27, 28, and 40	79
Table 7.	Comparison of commercial multibody simulation software packages	87
Table 8.	Synchronization of transitions through explicit and implicit events.	99

1 INTRODUCTION

1.1 Simulating the use of products: Setting the stage

An important goal of user-centred design is designing consumer durables for optimal interaction with users, i.e., considering how they can be used by various users in various situations. Possible forms of use can be investigated through usability testing, by relying on human subjects and physical prototypes, but this takes a lot of time and resources [1, p. 117]. Furthermore, in the case of conceptual design, adequate physical prototypes are often not available, and this makes real-life testing even more problematic. Under these circumstances, simulations with virtual prototypes have proven an attractive and often more efficient solution.

To achieve this goal, concurrent simulations of the physical behavioural and interaction processes are needed. While recently the first multiphysics behavioural simulation tools have become available for application in engineering, less advancement has been achieved in the field of simulating human-product interaction and complex use processes under varying circumstances. Currently, product designers do not have intuitive tools by which they can specify and control behavioural and interaction simulation processes in the conceptual phase of product design [2]. The work described in this thesis aims to close this gap and to offer a solution for the general problem described above.

With special attention to the conceptual design of consumer durables, this thesis proposes a theory for modelling, specification, and simulation of interaction processes during the use of a product. The theory comprises (i) a modelling approach based on the concept of *nucleus* that can be used for computer-supported behavioural simulation, (ii) an extension of nucleus-based modelling called *resource integrated modelling* that allows the specification of control mechanisms for behavioural simulation, and, ultimately, (iii) a specific implementation of resource integrated modelling for control mechanisms of human-artefact interaction, which is built around the novel concept of *scenario bundles* that is introduced in this thesis.

A scenario bundle is a formalized description of a set of alternative scenarios of the use of a product. Scenario-based simulations can provide designers with quantitative feedback on complete interaction sequences that may happen during product usage. By doing so, they provide clues on how designers can improve interactions with products. With a scenario bundle, a designer can perform ‘what-if’ type of studies involving variations of the product’s design, its physical properties, the surroundings of use, and human users. Each time when the arrangement is varied, the interaction simulation may take a different course through the scenario bundle. I have considered this kind of instrument as a useful complement to conventional engineering simulations, which require dedi-

cated simulation runs for each specific interaction situation occurring during the use process. Additionally, scenario-based simulation aims to offer a low-threshold alternative to testing physical prototypes or conducting interactive (participatory) simulations by eliminating the need for human subjects.

Originating in software engineering [3], the concept of scenarios has become widely used in various application fields of design. Informal scenario descriptions (also known as storyboards or use cases) have become a popular means in product design to explore possible ways of, and communication of preliminary ideas about, product usage [4-6]. Formalized scenarios have been used in computer-based simulations of human-artefact systems as well [7]. However, these are typically based on drastically simplified models. In the context of this thesis, a scenario has been defined as a specification of possible ways of using (i.e. conducting interaction with) a given product to accomplish some functions required by the users in a given surrounding [8]. The 'way' of using a product is associated with alternative human decisions influencing the course of the use process. Actually, these decisions define how the user interaction with the product manifests. In order to operationalize scenarios, the result of this mental control process must be included in the simulation. Scenario-based simulations may point out ways of using the product that may lead to some form of failure.

Figure 1 shows the key concepts of my approach and the activities the designer is supposed to perform using a full-fledged system based upon the foundations in the theory presented in this thesis. After having created or developed an artefact model for the product (optionally extended with logical control instructions for its embedded software), the designer can select a human model from a library, and he can conjecture up a scenario, or a bundle of interrelated scenarios of human-product interaction. The result is an interconnected composition of models and specifications, with which simulations of use processes can be performed and investigated. To explore (i) design variations, (ii) other forms of use, and (iii) use by humans with different characteristics, the designer can (i) vary the product model, (ii) 'play' with the scenario bundle, and (iii) insert other human models.

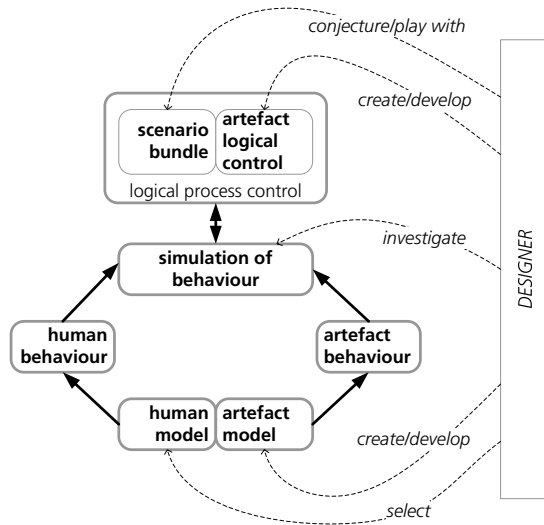


Figure 1. Key concepts of the approach presented in this thesis

1.2 Open issues in simulation of use processes

It has been argued in the previous subchapter that designers need means to investigate and evaluate the use of products during conceptual design without the involvement of physical prototypes and human subjects. Simulation could be a solution, but the current approaches are too specialized to cover all relevant aspects of use. Simulation has been defined as an experiment performed on models, typically through numerical evaluation on a computer [9,10]. Many different aspects of use are supported by different simulation approaches, but they cannot be deployed together to obtain a holistic simulation of use. To achieve any progress in this direction, researchers and developers should address three issues.

The first issue is diversity. As is shown in Figure 2, the different simulation approaches use different representation principles for the underlying simulation constructs (i.e., simulation models and simulation specifications). Moreover, (i) the simulation approaches have been orientated to the reconstruction of different behaviours, and (ii) different alternative algorithms are typically available for the prediction of a specific type of behaviour. As a consequence, it is difficult to create unified models that would allow for concurrent investigation of multiple behaviours influencing each other.

The second issue is how to consider use as a sequence of interactions. In interactive simulations, human subjects can recurrently interact with the computed process through some kind of control interface. However, no method or tool exists to include human control interventions in simulations without the involvement of human subjects. Conventional engineering simulation systems compute behaviour based on one initial set of data and conditions. This initial set is considered to be valid for the whole simulation process, either as fixed values from the beginning or as changing input signals that should be available in a predefined form from the start of the simulation. Such a set of data defines stimuli that are usually perceived as one interaction between human and product. Under this assumption, investigation of 'holistic' use processes as sequences of interactions is not possible.

The third issue is how to generate data corresponding to human control interventions in a simulation system. In real-life use

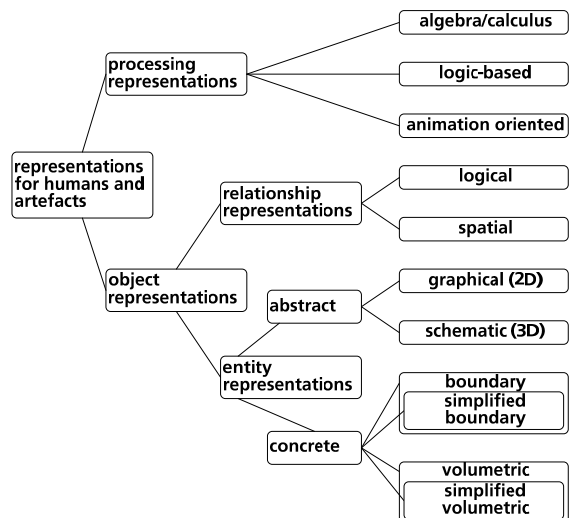


Figure 2. Representations in use for various simulation approaches

processes and in interactive simulations, the interventions are based on processing of sensory information by the human brain, which leads to decisions to activate muscles. Comprehensive simulation of the human senses and, especially, of the related actions of the human brain is however generally considered a challenge to science for decennia to come. It can be concluded that simulation of 'holistic' use processes without human subjects needs something that substitutes the control feedback loop through the human brain.

1.3 Research problems

From the issues described in the previous subchapter, it appears that the current behavioural simulation approaches that have become a common means of support in engineering fail to enable simulations of multifaceted use processes. To solve the problem, these simulation approaches need to be either replaced or extended. The solution should address the following concrete problems ensuing from the open issues:

- Current behavioural simulation systems still offer only limited support for multiphysics simulations. Although some multiphysics simulation systems have come to the market in recent years, these typically offer the possibility to investigate only preselected combinations of two types of behaviour (e.g., mechanical deformation and heat flow), rather than combinations of a multitude of arbitrary types of behaviour.
- The typical computation schemes of behavioural simulations are not the most suitable for all the processes to be simulated in the use of products. The problem is that they are based on knowledge about physics, which has been modelled with differential equations. If this principle would be used to include the workings of human perception and human cognition, it would mean that simulation algorithms have to be developed for the physics of neurological phenomena. It would also mean that these algorithms should include the physical processes that make elementary neural phenomena result in decision-making, and, finally, in control signals for muscles. Neurologists however have assumed that the signals involved are discrete rather than continuous [11,12], which implies that they cannot be simulated with differential equations [13].
- Some alternative simulation approaches based on discrete logic for human decision-making have been developed in the fields of psychology [14] and game theory [15]. However, these have a limited focus on finding the best strategy for solving a problem (or winning a game), in which each decision has its predefined set of direct results that do not depend on intricate interaction processes. As a consequence, these approaches address neither perception nor activation of muscles.
- If behavioural simulation algorithms are used for those processes in the use of products that can be modelled with differential equations, something is needed to specify or generate the changes in input data over time.

- Designers see it as their competence to anticipate product use, based on intuition and professional experience [16]. It means that they may have conjectures on how the human user of a product will react by reasoning about the signals he perceives during interaction. It seems an attractive option to use these conjectures instead of (the unavailable) behavioural models of perception and cognition. However, these are typically specified using informal scenario-building techniques and they are not available as processable algorithms.

1.4 Scope of this research, targeted deliverables, and criteria

The work reported in this thesis aimed to contribute to the Knowledge-Intensive Conceptualization (KIC) programme of the Section of Computer-Aided Design Engineering (CADE). This was one of the main research orientations for the section defined in the past research portfolio of the Faculty of Industrial Design Engineering. The KIC programme aspired to offer applicable solutions for product conceptualization and remote collaboration in the industry. Research projects in the programme dealt with topics such as hand-motion based input for shape design [17], and output and visualization technologies such as augmented prototyping [18] and holographic imaging [19]. Closely connected to the work in this thesis is the work that has been done in KIC in the areas of human body modelling [20,21], interactive simulation of human grasping [22], and particle-based artefact modelling with vague discrete intervals [23].

For the work in this thesis, my aim has been to devise a novel approach for simulation of manipulative interactions with products in order to support designers who want to test concept versions of products with virtual users. To offer that functionality, a software system is needed that (i) offers a designer-friendly user interface for modelling and specification of humans, products, and conjectured use processes as well as for running controlled simulations, and (ii) is optimized for computational performance, so that designers can assess its usefulness in practical settings. In the framework of this PhD research, only a proof-of-concept prototyping of the system has been planned, for two reasons. Firstly, the real novelty of the approach is in the underpinning theory, which introduces the principles of resource integration and scenario bundle-based simulation. In the second place, constraints in available time and resources had to be faced. Therefore, the research reported in this thesis focused on the theory, and on producing sufficient evidence to show that (i) conventional theories and approaches are not able to provide the requested functionality, and (ii) the novel theory can be converted into a structured set of processable algorithms that can serve as a starting point for further development, and that is realized in a proof-of-concept implementation. Furthermore, the objective has been to (iii) demonstrate the application of the algorithms to simulation of product use, (iv) investigate the limitations of applicability of the theory, (v) validate the theory for its functional affordances and its potential convenience of use from technical aspects, and (vi) provide di-

rections for future research and development towards a full-fledged system.

From the above scoping it can be derived that the main deliverables in terms of novel contributions to design support have been a *theory* and a *proof-of-concept implementation*. These have been elaborated with the following initial criteria in mind:

- The theory should provide a unified representation that allows designers to model and specify entities, relations, and processes with a minimum number of different types of elements;
- To support designers in the investigation of use processes, the unified representation must enable controlled simulation (i.e., concurrent simulation of behaviours and execution of specifications of conjectured interactions) based on processable algorithms – in other words, the core of the theory must consist of formal definitions;
- The results of controlled simulation should be available to designers in a form that gives comprehensive insight and is easy to interpret (among other things, it should be visualized through animation), both during and after simulation;
- In a way that does not necessitate simulation of human decision-making, the theory must explain how the use of a product can be considered as a sequence or network of connected interactions based on bundled sets of alternative scenarios;
- It must be possible to designate partial models and specifications for purposeful reuse in other investigations of use processes
- In typical cases, the system should not burden the designer with more preparation efforts and waiting time than are needed when working with conventional approaches.
- The research questions and hypotheses have been formulated with these criteria in mind. They are discussed in the next two sections.

1.5 Research questions

Related to the research problems identified in 1.3, I formulated five initial research questions to guide my research work:

(i) How can multiple areas of physics be simulated concurrently?

(ii) Is it possible to simulate arbitrary behaviours using just one generic modelling principle for entities (artefacts) and their behaviours?

These two questions address the problem that the numerous existing behavioural simulation approaches rely on many different models. Usually, if two or more different types of behaviour (e.g., mechanical deformation and kinematics) need to be investigated, they must be simulated with different models. Although it may be possible for the designer to convert his product model to each different

model that is required, it is generally not possible to simulate the corresponding behaviours concurrently, while taking into account all the mutual interdependencies. A modelling principle that can be used for concurrent simulation of arbitrary behaviours would resolve this problem.

(iii) How can designers use their conjectures about interaction processes to:

- (a) specify changes in simulation input data over time*
- (b) devise specifications that substitute simulation of the reasoning and decision-making in human users that leads to these changes*

Informal scenario building and storyboarding have become popular techniques for designers in the exploration of possible future use processes. It can be said that these scenarios and storyboards contain conjectures about human control of use processes – in particular about the feedback loop through the human brain. However, the records of applying these informal techniques cannot be linked to simulations. To make that possible, they must be available as a formalized specification.

(iv) How can the models and specifications needed to simulate use processes be connected and arranged in a way that enables computational processing, and at the same time allows reasoning about human-product interactions?

The previous questions point to solution elements in the form of simulation models and interaction specifications. These must be computationally processed together, which requires a connection and arrangement scheme. Also, as a whole, these connected models and specifications together should be easily interpretable to designers. Hence it is desirable that the models and specifications are connected and arranged in a way that corresponds to how real human-artefact systems are connected and arranged.

(v) How can a concept of a system to support use-process simulations be organized to facilitate modular development of building blocks supporting distinctive design activities?

This question addresses a practical aspect of conducting the research work. As was indicated in 1.4, it was not possible to develop and test a full-fledged system within the framework of the PhD research. The available time and resources can be used more efficiently by focusing prototype system development on building blocks supporting those design activities that specifically utilize the novel functionality of the system. More specifically, it is assumed that supporting the design activities of *specifying conjectures about interactions*, and *deploying these specifications to enable simulation of interconnected interactions* are to be prioritized over instantiation of artefact models and human models, and performing behavioural simulations of single interactions. In particular, this makes it possible to decouple the work reported in this thesis from ongoing research in the CADE sec-

tion related to nucleus-based modelling and simulation.

1.6 Research postulate and hypotheses

Based on the research questions in 1.5, hypotheses were formulated to serve as a starting point for development of the theory and proof-of-concept implementation and for the identification of available means to justify the achievements. As it emerges from the criteria in 1.4, support for multiphysics simulations (research questions (i) and (ii)) was not comprehensively addressed in the elaboration of the deliverables. This decision was based on (ii) the aforementioned constraints in available time and resources, and (ii) the fact that other research in the CADE section had already produced a theory that aims to solve this problem. Therefore, the veracity of the available theory describing *nucleus-based modelling and simulation* [24] was taken for granted as a research postulate rather than as a hypothesis:

***Postulate:** To enable integrated simulation of arbitrary physical behaviours, nucleus-based models can be used. By explicitly managing relations between the entities, nucleus-based models extend the scope of modelling so as to allow us to specify multiple types of behaviour ('multiphysics') and to perform behavioural simulations.*

The postulate specifically addresses research questions (i) and (ii).

The other research questions could be addressed by four hypotheses:

***Hypothesis 1:** Additional relations, so-called change relations grouped together into transitions, can be defined according to the principles of nucleus-based modelling to specify the connections between interactions in time.*

This hypothesis addresses research question (iii)(a) and the application of the generic modelling principle addressed by research question (ii) outside the domain of multiphysics modelling.

***Hypothesis 2:** By allowing the specification of (iii) logistics as a third layer (Figure 3) above (i) behavioural simulation and (ii) nucleus-based modelling, transitions enable simulation of subsequent and interconnected interactions. The resource-integrated models built with these three layers permit prescribing, organizing, and connecting procedurally disjunct sequences of interactions based on the principles of nucleus-based modelling.*

This hypothesis addresses research question (v), and also (ii) and (iii)(a).

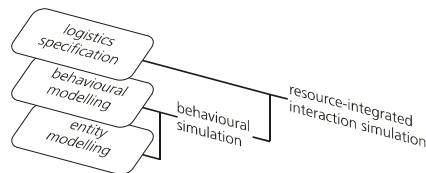


Figure 3. Three-layered concept of modelling, simulation, and logistic control of simulations.

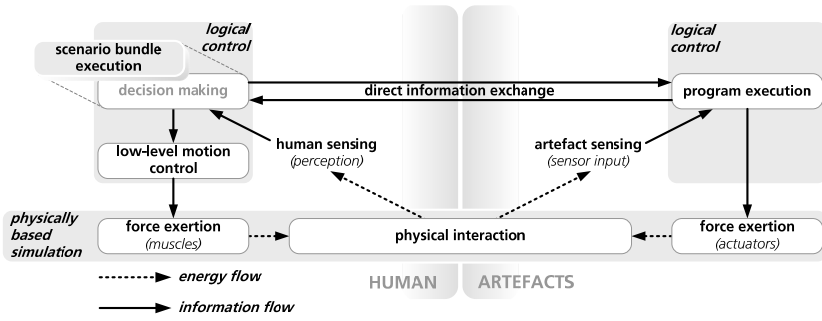


Figure 4. Conceptual reasoning model of human-artefact interaction

Hypothesis 3: As a specific manifestation of resource integration, scenario bundles can be used to specify transition relations representing human interactions. A scenario is a possible way for a human user to control his interactions with a given product in given circumstances. Bundling allows designers to create organized sets of scenarios they have conjectured.

This hypothesis addresses research question (iii)(b).

Hypothesis 4: A simplified reasoning model of human-artefact interaction, shown in Figure 4, can be used to bring the aforementioned concepts together into a comprehensive use-process simulation approach. This reasoning model arranges and connects the constructs, models, and specifications used in a scenario bundle-based simulation as well as the functions performed by humans and artefacts.

This hypothesis addresses research question (iv).

1.7 Research methods and structure of the thesis

The work reported in this thesis has followed the nine-stage scheme of design-inclusive research [e.g., 25] as is shown in Figure 5. First, an explorative study was carried out to establish the current state of the art in simulation of processes and behaviours with a view to the use of products. In the discourse below, the specific *methods* (in *italics*) that have been applied are discussed for each stage.

The two key research methods that have been applied in exploration were *critical literature review* of academic achievements and *query-based web search* to enable examination of commercially available systems. These methods have been applied to establish the state of the art in (i) what is to be expected from cutting-edge simulation approaches currently being studied in academia and (ii) what is already available to designers in the form of off-the-shelf and configurable software systems. The emphasis in the literature was twofold. Firstly, behavioural simulation¹ approaches were reviewed on their capabilities to support in-

¹ behavioural simulations involve processes governed by the laws of nature

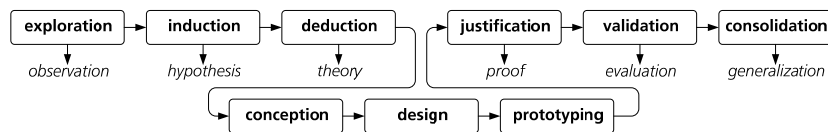


Figure 5. Nine-stage scheme of design-inclusive research

investigation of processes in user product interaction, and secondly, approaches for control of simulations were reviewed with a view to their abilities to support scenario-based control of behavioural processes. To assess simulation capabilities, it was also necessary to derive classification schemes of behaviours according to what is generally agreed upon in the literature. In that respect it was not only necessary to consider the common areas of physics (mechanics, thermodynamics, etc.) as distinctive behaviours, but also biological functions of humans (perception, cognition, etc.). Critical examination of simulation approaches revealed the opportunities offered by (combinations of) available approaches, and also the deficiencies. The exploratory phase has been reported in Chapter 2 of the thesis.

Logical induction was the research method applied to the insights obtained from the exploration in order to derive the research problems, questions, and hypotheses presented in the previous subchapters, 1.3-1.6.

The theory in Chapter 3 was deduced by applying methods of *propositional logic*, and also by applying methods of *theory adaptation* and *theory integration* to the already existing theories of (i) nucleus-based modelling, (ii) discrete event systems, and (iii) finite automata. These theories had emerged from the exploration phase as useful building blocks for the requested novel functionality.

To provide supporting evidence for the justification and validation of the theory and to confirm that it could be converted into a structured set of processable algorithms, it was implemented by *developing a proof-of-concept system* as reported in Chapter 4. This implementation, which was realized by *deployment of proxies* for the functional elements of the ultimate simulation system, involved *adaptation of configurable and programmable software components* from commercial vendors. The proof-of-concept implementation has been submitted to *experimental application and testing* in Chapter 5. This was had been set up by applying a procedure of *component-based application development*. First the proof-of-concept implementation has been tested to simulate elementary interactions as individual components, and then the components have been combined to enable composite simulations of interactions with products.

In Chapter 6 it is shown how the theory was justified by submitting its elements to *evidential reasoning with consequences*. By applying *commonsensical argumentation*, this research method shows where the theory can be accepted as empirically correct (proper), where it is partially correct, and where it fails. Since experimental application and testing had produced only a small number of application cases for which the theory was shown to be correct, the emphasis has been on revealing the consequences of the theory in terms of its limitations. In that sense, the verification mainly amounted to partial *falsification* of the theory.

In Chapter 7 the theory and the approach were validated by *comparison with a reference approach* that would offer designers the possibility to interconnect simulations by using a conventional simulation system. *Commonsensical argumentation based on facts* was used to show that the reference approach is inferior with respect to critical parts of the requested functionality. Comparison with the reference approach was also used to assess the convenience of use from the technical aspects ‘required preparation effort’ and ‘time needed for simulation runs’. This comparison provided the opportunity to enter the generalization stage. It could be realized by applying *abstraction* to reasoning about preparation steps and the complexity of application cases, which could be used to derive generally applicable statements about convenience of use through *mathematical deduction*.

Chapter 8 further generalizes the merits and shortcomings of the presented research work in this thesis by *drawing conclusions* based on the findings, and proposing directions for future research, which have been derived by *analysis of critical elements the shortcomings*.

1.8 Forerunning publications

In the years 2003-2008, parts of the research work reported in this thesis have been published in conference proceedings. A comprehensive review of engineering simulation approaches was presented in [26]. The approaches in this survey concentrated on behaviours of artefacts, and also control of artefact simulations. An extension that includes human behaviours and human control in simulation approaches was presented in [27]. Together these knowledge exploration papers clarified the research problems and provided clues for solution elements that could be used for the theory. An introduction to the theory of nucleus-based modelling has been presented in [28]. A formal elaboration of the nucleus theory with emphasis on the data structure of models was presented in [24]. Initial work towards application to use processes, including an early pilot application that involved scheduling based on a basic scenario, was presented in [29] and further elaborated in [30]. A first proof-of-concept implementation with scenario bundles (named differently at that time) was tested by simulating a use process with two basic interactions in [31]. The proof-of-concept implementation presented in this thesis, and experimental testing by simulating four basic interactions combined into a composite interaction process, was presented in two papers. In the first of these two, [32], building and inclusion of nucleus-based models of artefacts and humans was elaborated. The second paper [33] focused on the specification of scenario bundles.

In the years 1999-2003, the way to the research in this thesis was paved by studies on using formal representations to specify product life-cycle processes (one of which is the use process). Looking back, the representation concepts published in these years can be considered predecessors of scenario bundles, although the label ‘scenarios’ first emerged when the focus of research was narrowed down to use processes in 2002. The overview of related publications is

incomplete without mentioning some anticipating work reported in conference and journal papers. In [34], process trees, of which the use part in fact represents a scenario, were proposed to specify the product life cycle. In [35], a set theory-based representation scheme for sequenced, connected episodes of continuous processes (which might have been simulation results) was presented. A first survey of use-process representations, which include some of the control schemes surveyed in the thesis and a first proposal for linking with simulations was presented in [36]. Finally, in cooperation with the Mizoguchi Lab of Osaka University, a way of scheduling user interactions (tasks) together with product functions based on ontologies was proposed in [37].

2 KNOWLEDGE EXPLORATION AND LITERATURE STUDY

2.1 General introduction

Simulation has been defined as an experiment performed on a model [9]. Simulation approaches can be characterized by (i) the models they are based upon and (ii) the types of experiments that are conducted. The goal of this review is twofold. The first goal is to establish the state of the art in behavioural simulation regarding (i) the capabilities to support the variety of behaviours and processes related to the use of products and (ii) the practical deployability in a context of product design, with designers who use 3D modelling tools for products, and who prefer easy-to-interpret simulation outputs, such as animations. The second goal is to establish the state of the art in logical control of simulations with a view to application of scenarios to simulated use processes. As will be shown below, the first objective appeals to the *simulation models* mentioned in the definition and the second objective appeals to a particular type of simulation experiment being of specific interest to this research work.

A simulation model represents assumptions of how the real-world system of interest behaves, and it usually takes the form of mathematical or logical relationships [10], although it can also be a physical prototype or mock-up. The work reported in this thesis focuses on virtual prototyping for use processes, which is typically based on digital models consisting of mathematical and/or logical relationships. These relationships hold knowledge about *behaviours* that can be observed in human-product-surroundings systems in the real world. Simulation approaches are strongly associated with a choice for a particular (combination of) model type(s). The capabilities and the deployability of a simulation approach depend on these models.

A simulation of a use process can only achieve a certain degree of comprehensiveness if the models that are used allow inclusion of behavioural knowledge of that degree of comprehensiveness. Also, simulations can only be used for particular practical purposes (for instance, embedding of 3D models from a CAD system) if the models allow it. In the first part of this survey (Subchapter 2.3) *behavioural* computer simulations of processes related to use have thus been assessed based on the (combinations of) models that they rely upon.

Experiments are typically conducted by administering stimuli to some kind of system², and observing the responses of that system to those stimuli. In simulation, classes of experiments can be distinguished based on the possible schemes of how stimuli are administered. Another word for ‘to administer stimuli’ to simulations is to *control* simulations. There are two principal schemes of control: open-loop control and closed-loop control [38]. If applied to simulations, closed-loop control implies that the administered stimuli depend on outputs of the simulation, while in open-loop simulations they do not. Examples of open-loop simulations are (i) so-called batch simulations that apply a set of initial stimuli at the start of a simulation run only [39], and (ii) simulations applying stochastic dosage of stimuli during runtime, for instance by using random number generators [10]. For closed-loop controlled simulations, three main principles have been distinguished: (i) interactive or *human-in-the-loop simulation*, where a human subject interacts with the simulation to control it, (ii) *hardware-in-the-loop simulation*, where a hardware device (for instance a physical prototype) interacts with the simulation, and (iii) *software-in-the-loop simulation*, where the control has been programmed into a piece of software that is connected to the simulation software [39,40].

Since my objective has been (i) to include human interactions in simulations, and also (ii) to make designers independent from human subjects and physical prototypes, the forms of simulation control that are reviewed in Subchapter 2.5 are closed-loop, and more specifically *software-in-the loop* approaches. My particular focus has been on the various schemes and representation forms that can be used to specify control as *scenarios* describing possible ways of using products. However, from a practical viewpoint, a consistent thematic arrangement of topics in the review was easier to achieve by discussing all control-related aspects of simulations in one subchapter. Consequentially, the review in Subchapter 2.5 does not only address scenario-based control of simulations, but also other approaches to control of simulations and also *simulations of control*. The latter addition was made because simulation of control and control of simulation typically use the same algorithms and because the distinction between the two is rarely made in the literature. The review in 2.5 thus also covers behavioural simulations of control mechanisms, which were therefore omitted from 2.3.

2.2 Introduction to behavioural simulations

2.2.1 Considerations in reviewing behavioural simulation approaches.

Typically, the models that are used to build simulation constructs are behavioural models, i.e., they hold knowledge about particular behaviours of entities. Such a model represents assumptions of how the real-world system of interest behaves,

² in simulations, the *system* is a simulation model of a real system

and it usually takes the form of mathematical or logical relationships [10]. Types of behavioural models vary from two perspectives, namely, (i) the form of representation (for instance, as a 3D geometric model or as a set of equations) and (ii) the entities they can represent (for instance, vehicles, computer systems, or organizations). To structure my review of behavioural simulation constructs I have used these two perspectives. In the first place, I subdivided simulation approaches based on the representations used in simulation models. In 2.2.2 I will introduce the classification scheme for virtual representations that was used for further subdivision. In the second place, I arranged my review of each simulation construct according to the entities for which it has been used as a simulation representation.

The review was confined to two types of entities commonly distinguished in simulations, namely, artefacts and humans. Together, simulation constructs for these two entities allow consideration of humans, products, and artefacts in the surroundings, but the confinement excludes non-artefactual elements surrounding the human and the product.

By not reviewing simulation approaches for these elements I have ignored the potential to simulate behaviours of (i) the ambient (or climatic) circumstances that surround the human and the product, and (ii) natural entities such as animals, plants and terrain. The ambient circumstances of use include lighting, heating, and the air with its temperature, pressure, humidity, draft/wind, and precipitation.

Climate simulation belongs to the domains of work place design and architecture (indoor climate) and meteorology (outdoor climate). Ignoring climatic simulation appears to be acceptable if the climate during use is considered a constant condition rather than a fluctuating influence that needs to be computationally evaluated. Natural entities can be ignored if it is taken for granted that in the use of many products (especially products for indoor use) these do not play a role. Therefore I have restricted modelling and simulation of the surroundings of the product and its human user to 'other artefacts'. In each of sections 2.3.1-2.3.8 of the detailed review, approaches for artefact simulation, human simulation, and combined human-artefact simulation have been explicitly addressed.

2.2.2 Classification of simulation constructs and representations

As was mentioned above, the discussion of the various approaches to behavioural simulation has been structured based on the theoretical basis of the processable constructs that have been used for simulation. The right-hand side of Figure 6 gives an overview of the constructs that have been distinguished in this survey. The taxonomy at the left-hand side is identical to Figure 2. It shows how the constructs have been derived from common representation forms. At the highest level of this taxonomy, processing representations and object representations are distinguished. They represent some form of processing associated with the artefact (typically an interpretation of its behaviour) or the artefact itself, respectively. Processing representations can be subdivided into algebraic/calculus-based representations and logic-based representations. Combinations of these

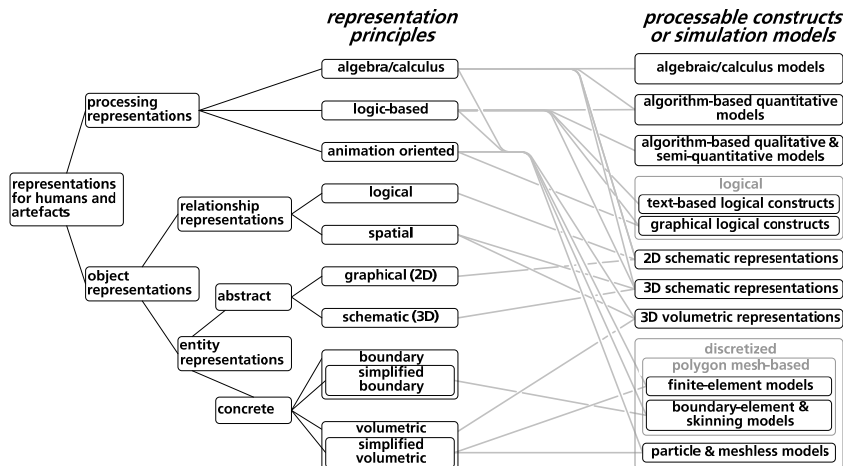


Figure 6. Taxonomy of model types used in simulation

two are known as algorithms. The survey distinguishes algorithms that support visualization of changes over time during processing as ‘animation oriented’.

There are two types of object representations: relationship representations and entity representations. Relationship representations describe logical or spatial relations. Entity representations describe arrangements and/or shapes of entities either in an abstract or concrete way. Abstract entity representations are based on 2D-graphics or 3D-schematics. Concrete entity representations are typically boundary representations, simplified boundary representations, volumetric representations or simplified volumetric representations.

The right-hand side of Figure 6 shows that most simulation constructs are based on combinations of representation types. The discussion of the individual simulation has been arranged according to these common combinations.

2.2.3 Objectives, scope, and assessment criteria

The survey focused on research achievements published in scientific literature but relevant commercial solutions have also been included; these have been referenced in footnotes³. Priority was given to achievements and examples related to the use of consumer durables. Contributions from other fields have been included if no search results could be obtained from the focus area.

The various approaches to simulation have been assessed based on their potential contribution to investigation of use processes. The following criteria have been considered:

- *Range of behaviours covered.* Simulation should cover as many types of human and artefact behaviour as is reasonably possible. In the next two sections, schemes will be introduced to classify the various behaviours for artefacts and humans, respectively.

³ Most of these footnotes refer to websites. Links that have become obsolete after the publication date of this thesis may possibly still be available through www.archive.org.

- *Relevance of the scope.* The overlap between the scope of a simulation approach and the scope of the application area, use of consumer durables, should be as large as possible.
- *Ease of preparation.* The amount of time needed to set up a simulation should be kept at a minimum. In particular, this concerns inclusion in the simulation of product models that designers have created. My premise has been that most designers of consumer durables use solid-modelling CAD packages. If these models cannot directly be used as a virtual prototype, a second-best option is that available CAD models can be *converted* to simulation models in an automated way.
- *Speed and computability.* The time needed for a simulation to run on common hardware should be as short as possible.
- *Ease of interpretation.* Traditionally simulation output is numerical, e.g., tables or graphs show the course of values in time. My assumption is that, especially to designers, spatial 3D animation of the simulated system is a valuable addition to numerical output.
- *Fidelity of the outcomes.* The outcomes of the simulation must sufficiently correspond to real behaviour.
- Since no simulation approach covers all the aspects of use, it is also worthwhile to consider to what extent the *availability of combination options* and the *exchangeability of data* between simulation approaches allows further extension of the scope. However, a comprehensive assessment of these options for N approaches would involve investigation of all N -to- N combination possibilities. Instead I have followed a more pragmatic strategy by assessing *existing* combinations of simulation approaches (for instance, combinations of mesh-based models with volumetric 3D models) that I found in the literature and in commercial documentation.

2.2.4 Types of behaviour in artefact simulation

- Usually, the possible artefact behaviours are classified according to the common areas of physics: mechanical behaviour, acoustic behaviour, optical behaviour, etc. (Figure 7). The effects of these behaviours can be observed as flows and transformations of energy and matter. To enable simulations the knowledge about these behaviours is usually modelled as systems of differential equations, which are continuously evaluated over time, hence the name *continuous simulation* [13]. As will be further discussed in 2.4 and 2.5, certain control mechanisms are also simulated based on these principles, because the signals in these systems directly correspond to physical phenomena.

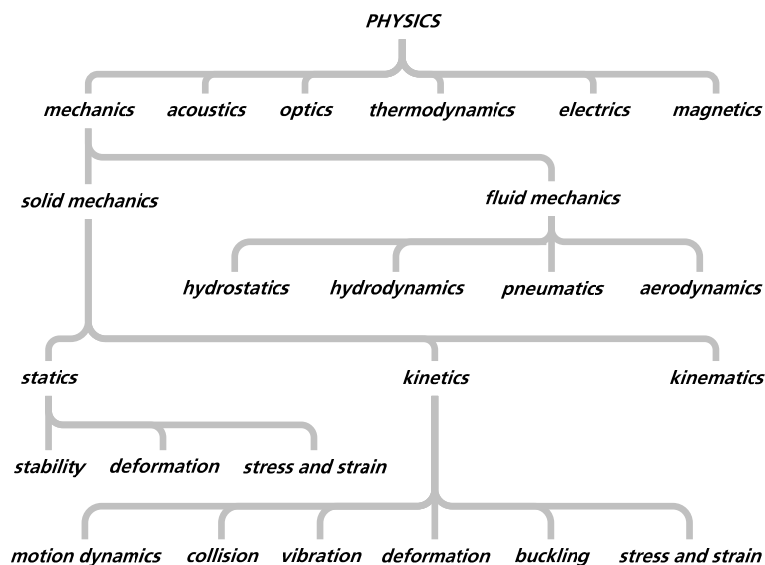


Figure 7. Taxonomy of the areas of physics

2.2.5 Types of behaviour in human simulation

I have typified human behaviours starting with the simplified general reasoning model shown in Figure 8. It is based on elementary functions that can be recognized in the human body relating to the processing of matter, energy and information. In this general reasoning model I have distinguished metabolic behaviour, perceptual behaviour, cognitive behaviour, control behaviour, actuator behaviour, kinetic behaviour, kinematical behaviour, and non-mechanical physical behaviour (not shown in the figure).

Metabolic behaviour involves the chemical processes that convert matter taken in as food, drink and air to energy [41]. My assumption has been, that the relevance of metabolic behaviour to product use is limited to (i) aspects of human endurance and fatigue, as they are investigated in the design and development of military equipment [42], work environments [43], and sports equipment [44], and (ii) aspects of ingesting and processing consumables [45] rather than durables. Since these particularities do generally not apply to consumer durables, metabolic simulations have been disregarded in this survey. Perceptual behaviour is the behaviour of sensory cells and systems that convey the impingement of electromagnetic, mechanical, and chemical changes in energy [46], which is passed on by the central nervous system to the brain.

Cognitive behaviour relates to information processing by the brain, which includes attention, remembering, producing and understanding language, solving problems, and making decisions [47]. Control behaviour involves conscious and subconscious control of human action [48]. Together, perceptual behaviour, cognitive behaviour, and control behaviour make up the human control loop, which is further explained in the introduction to the review of control of simula-

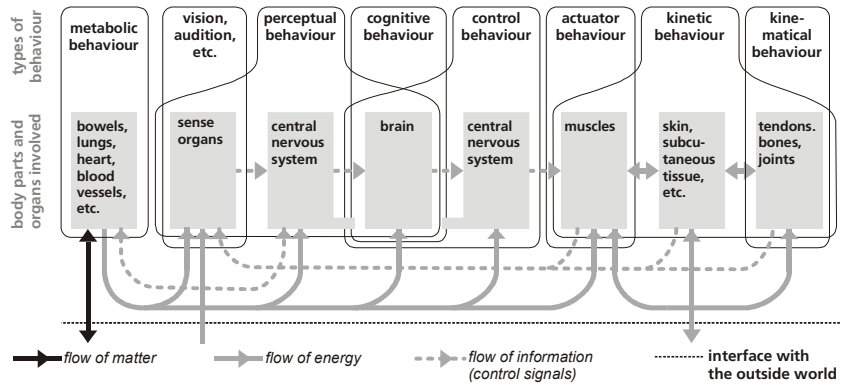


Figure 8. Typification of human behaviour based on elementary functions, including the relations with the involved body parts and organs.

tions in 2.4.3.

The next main category of behaviour in Figure 8, *actuator behaviour*, relates to human-initiated output of mechanical energy, i.e., producing motion at joints through contraction and relaxation of muscles [49]. *Kinetic behaviour* involves all mechanical behaviour concerned with the effects of forces on the motion of body parts and other objects, with the exception of actuator behaviour performed by muscles⁴. Typical body parts involved in kinetic behaviour are the skin, subcutaneous tissue, the skeleton and muscles. Although traditional rigid-body kinetics can be applied to certain parts of the human body (in particular, the bones), large deformations typically play a crucial role [50].

Finally, *kinematical behaviour* relates to the movement of bones and joints caused by muscles and external influences, without consideration of the energy aspects. Also included in the review is non-mechanical physical behaviour that can be subdivided according to the areas of physics as in Figure 7. It has been left out of Figure 8 to reduce its complexity.

2.2.6 Structure of the review

The structure of the evaluation of behavioural simulations in 2.3 is as follows. In each of the sections 2.3.1-2.3.8, a group of simulation approaches has been evaluated, following the subdivision on the right-hand side of Figure 6. First, a general description has been given (i) to characterize each approach based on its fundamental principles and (ii) to summarize the *range of behaviours*⁵ that can be simulated. This initial assessment of behaviours has mostly concerned the coverage of areas of physics according to Figure 7. Then, a selection of software and tools has been listed. Some of these are available from commercial vendors while others are still under development in academia. Subsequently, examples of use process-related applications to simulation of artefacts, humans, and human-

⁴ Statics is treated as a special case of kinetics with zero resultant forces.

⁵ Text in *italics* refers to the evaluation criteria listed in 2.2.3.

artefact systems have been included to address the *relevance of the scope* and to assess the *range of human behaviours* according to Figure 8. After this descriptive part, each section concludes with a critical assessment of the approaches addressing (i) general advantages and disadvantages in comparison with comparable other approaches, (ii) the *ease of preparation of simulations*, and (iii) the *ease of interpretation of the results*. From a practical point of view, it turned out that of the other criteria listed in 2.2.3, the range of behaviours covered and the relevance of the scope could better be included in the foregoing descriptive part, while the remaining two criteria could not be evaluated. It was found that *speed and computability* and the *fidelity of the outcomes* could not be used for a fair comparison of the approaches. The literature sources appeared to report on simulations of different human-artefact (sub)systems of varying complexity. Also, some approaches are still under development while others have matured and are commercially available. These two factors made it difficult to compare the performance and the fidelity of the outcomes.

After the assessment of individual categories of approaches, the overall findings are discussed in 2.3.9.

2.3 Review of behavioural simulations of artefacts and humans in time

2.3.1 Simulations based on algebraic and calculus descriptions

The first group of simulation approaches is purely based on algebraic and calculus descriptions, i.e., the simulation model does not contain instructions for algorithmic processing. In the conventional calculus-based approach to simulation, sets of symbolic equations specify a particular situation or a class of situations to which laws of physics apply [51]. Usually the investigated system, the situation and the involved laws have been idealized to reduce computing time. The physical behaviour is predicted by solving differential equations in the time domain.

Predicting behaviour based on deriving and solving differential equations can be done completely manually. Computer support has been considered or is available for (i) deriving differential equations and boundary conditions based on given system descriptions, (ii) finding analytical solutions for given differential equations, (iii) solving differential equations numerically and (iv) calculating the values of system variables based on the time-dependent functions that form the solutions of the differential equations. Computer support to derive differential equations is based on automated derivation from object models or on catalogues. Gelsey [52] introduced a knowledge-based approach for automated derivation from object models which was able to derive differential equations for kinematical behaviour directly from CAD models. I could not find references to further developments of this approach.

Catalogue-based classification based on solution principles (i.e., subsystems that fulfil given functions) has been proposed, among others, by Roth [53]. In

these approaches, the computer merely offers the designer a database with equations to choose from but it does not solve them. To solve differential equations analytically, commercial software based on symbolic manipulation can be used, e.g., Maple [54], which is also able to calculate the course of system variables based on derived solutions of differential equations. The simulation output is typically numerical.

Many examples of application of this classical and widely accepted simulation approach to use processes are found in textbooks. For instance, in a textbook by Shigley [55] differential equations have been solved to predict the time a clutch needs to stop a rotating shaft and to calculate the generated heat flow. In [56] numerous other examples from the subfields of solid mechanics can be found. An example from a different field is the simulation of aeroacoustic behaviour of a vacuum-cleaner fan by Jeon et al. [57]. From the 1970s on, systems of equations from Newtonian dynamics have been applied to develop models of (parts of) humans in the area of biomechanics [58,59].

Numerical simulation output has been used as input data for the animation of 3D human models. Griffin [60] presented several models based on equations that describe the rigid-body kinetics of the human body as a mass-spring-damper system to investigate vibrations. Job et al. [61] applied algebraic models to investigate pathological phenomena in the eye to simulate the induction of electric current in the retina. Other examples of equation-based modelling of non-mechanical physical behaviour, namely emission and absorption of heat or electricity through the skin, can be found in papers by Mochnacki and Majchrzak [62] and Panescu et al. [63], respectively. Equation-based descriptions have also been applied to modelling and simulation of human-artefact systems. An example is the simulation of external impact on a human head covered with a helmet presented by Therrien and Bourassa [64].

Regarding the human-behaviour representations, the investigated models have in common that they are not explicitly limited to one particular human or a particular group of humans, except where behaviour of humans with specific disorders is concerned [e.g., 61]. Apparently, the models are assumed universally valid for all 'healthy' humans. Typically they contain variables for which the values can be changed, thus offering the necessary versatility in dealing with different individuals while the equation itself remains in the same form. This is different from the situation in artefact simulation, where a newly designed class of artefacts often requires new dedicated differential equations to model observed physical behaviour, which makes the preparation of simulations labour-intensive, unless the aforementioned automatic derivation from CAD models is applied.

The disadvantage of labour-intensive preparation also applies to human-simulation models that include artefacts [e.g., 64]. The output of algebraic-description based human simulation is typically numerical and therefore not easy interpretable. Direct 3D animation is only possible by manually defining the links between computer-generated output and an animatable object representation.

2.3.2 Algorithm-based quantitative simulation of continuous systems

Algorithms generally combine algebraic expressions with formal procedural logic that describes the process of computation to calculate values of simulation parameters and evaluate conditions that determine which algebraic expression is valid. Algorithms are usually formally defined by using a programming language such as C++ or SIMULA [65], a specification language such as XML or UML [66], or a combination thereof [67]. Other procedural languages that have been specifically developed for simulations can be found in the survey by Sinha et al. [68]. Algorithms can be used to solve differential equations numerically, for instance if no analytical solution exists. To that end, several methods have been developed, e.g., Newton-Raphson and Runge-Kutta [69]. They have been included in commercial mathematics software such as Maple, Mathematica and Matlab⁶.

In fact, most of the simulations discussed in 2.2.1 have used numerical approximations, but since the actual behavioural models are purely based on equations, I have not considered these models as primarily algorithm-based. The typical application of algorithms to the behavioural models themselves is to add logic dealing with behavioural laws that introduce discontinuities in the course of a process. This is the case when selection of the governing physical laws depends on conditions. A change in the set of involved laws causes a transition in the behaviour, for instance when objects collide in 3D space. Baraff [70] introduced an algorithm for fast computation of collision behaviour of rigid bodies. Hummel and Girod [71] presented an algorithm for colliding elastic flexible bodies. The application domain of these algorithms is typically artefact simulation. Apparently, the approach is less suitable for human simulations since the dominant involvement of soft tissues in human collisions eliminates the discontinuities.

Regarding ease of preparation and ease of interpretation of the outcomes, these purely algorithm-based approaches are comparable to the approaches discussed in 2.3.1. They do not offer support for conversion from CAD files and they typically produce numerical simulation output.

2.3.3 Algorithm-based qualitative and semi-quantitative simulation of continuous systems

Qualitative simulation is based on the theories of qualitative reasoning, qualitative physics and qualitative process theory [72,73]. It has been developed for the investigation of incomplete system models, predicting behaviour based on qualitative differential equations (QDEs). A QDE is an abstraction of an ordinary differential equation. It is qualitative because (i) it describes the values of variables ordinally (e.g. low – medium – high) rather than in numbers and (ii) relations between variables are described as monotonic functions (e.g., y decreases if t increases) rather than algebraic functions. These descriptions can be augmented with semi-quantitative bounding intervals [74]. Qualitative simulation is often

⁶ maplesoft.com, wolfram.com, mathworks.com

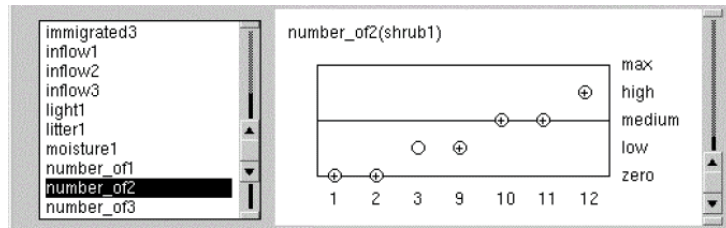


Figure 9. Output of qualitative simulation of vegetation growth with the simulation package VisiGARP

based on the qualitative reasoning theory [72] that can be used to create and simulate incomplete models.

Examples of software tools to support the creation and simulation of qualitative models are Qualsim and GARP. Qualsim has been criticized for its inability to solve non-trivial problems [75], and it seems that no publications have been referring to it since the early 1990s. The GARP architecture [76] offers a means to build models using a representation that is similar to the finite automata representations that are commonly used for control simulations, and that will be discussed in 2.5.2. Using an add-on module called VisiGARP, simulation results can be visualized (see Figure 9 as is discussed below).

Qualitative simulations have been applied to a wide range of physical phenomena appearing in artefacts. Kramer et al. [77] patented a method for qualitative simulation of kinematics in linkages. Bozzo et al. [78] have applied it to predict deformations in flexible beams. The only qualitative approaches to human simulations I found in the literature were a control simulation [79], and a metabolism simulation [80] that falls outside the scope of this review.

Just like quantitative differential equations, QDEs typically have to be drafted manually, and similarly, catalogues have been proposed to provide predefined QDEs for system components [81]. It is also possible to derive qualitative simulation models automatically from object models, in particular based on bond graphs [82] (see 2.3.4) but from what I could find in the literature, not from CAD models. Visualizing the output of qualitative simulations is difficult because of its qualitative nature. Figure 9 shows an example [83, reproduced with permission of the authors]. Another drawback of qualitative simulation is that for complex systems the simulation frequently is intractable or the model involves an expansive behavioural description that is hard to comprehend [84].

2.3.4 Simulations based on 2D graphical object representations: block-diagram and bond graph models

The simulation approaches in this section and in the following ones are primarily based on object representations, i.e., they represent artefacts in the first place.

Simulation approaches based on 2D graphical object representations are the ones based on block diagrams and on bond graphs. These are abstract 2D graphical entity models, with logical relations to define physical connections. Block diagrams are built up from predefined blocks that algorithmically represent physical laws describing the behaviour of components such as resistors, amplifi-

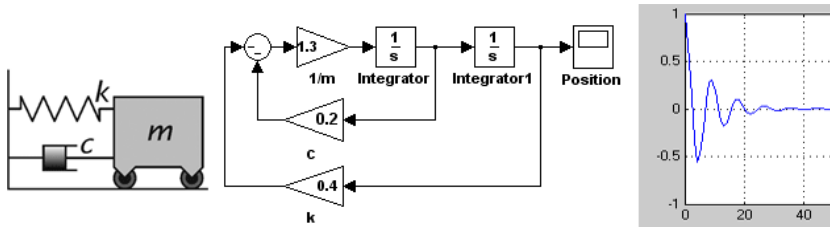


Figure 10. Mass-spring-damper system with Simulink block diagram and simulation output

ers, etc. Together with the relations, behaviour descriptions of components form an algorithm for simulation of the behaviour of the system. It is not a prerequisite that behaviour descriptions have a direct correspondence to components that can be physically distinguished.

Bond-graph based simulations have been introduced as an alternative to block diagrams by Paynter in the 1950s [85]. They can be converted to block diagrams [86]. Simulation is based on energy flows between elements representing components with basic physical characteristics [87]. Analogies between different areas of physics allow for using the same building blocks for mechanical, electrical, hydraulic, etc. components and perform multiphysics simulations [88].

Several commercial software tools are used to create and simulate block diagrams. Examples for general use are ACSL, Simulink, VisSim [89], and Dymola [90]. The latter package can also be used for bond graphs [91]. Various other software packages to create and simulate bond graph models can be found in the reviews by Montbrun-Di Filippo et al. [92] and by Samantaray⁷.

Block diagrams are mostly used to model and simulate signal-processing and control systems (see 2.5.1), but they have also been applied to mechanical systems [93]. Figure 10 shows a simple example⁸. Bond graphs have been applied to simulate processes related to the use of consumer durables. For example, Remmerswaal and Pacejka [94] used bond graphs to simulate forces during the handling of a vacuum cleaner. To enable bond-graph modelling of the human body, which is largely built up from continuum structures, it is typically simplified in order to make modelling as discrete components possible.

Margolis [95] developed a bond-graph model to simulate the response of the human body as a whole to external vibrations. To model and simulate musculoskeletal structure and function, Wojcik [96] proposed modular bond graphs. She had acknowledged that the appearance of the resulting models is rather cumbersome (Figure 11), but had nevertheless suggested that the use of bond graphs could promote standardization in human-body modelling. Pop et al. [97] applied bond graphs to create a partial model of the human body to simulate walking. Hubbard [98] presented a human-artefact bond-graph model of the use of a vaulting pole. The simulation includes kinematics and rigid kinetics of the

⁷bondgraphs.com/software.html

⁸adapted from Stanciu, tinyurl.com/79fza

whole human-artefact system, as well as dynamic deformations of the pole.

Regarding the ease of preparation, two-dimensional graphical object representations suffer from several issues. In conventional block diagrams, the relationships between blocks are defined as a unidirectional energy flow, which defines a procedural input-output treatment for the simulation computation. A disadvantage of this approach is that, although the diagram represents the object, there is no visual resemblance [88]⁹, as is illustrated in Figure 10 (page 24). Ferretti et al. [99] claim that, additionally, the flow through the blocks does not conform to natural human reasoning about physical behaviour. They advocated block modelling approaches such as Dymola, which are based on declarative rather than procedural relations. These result in block diagrams without causal arrows, in which the blocks have the same connections as the components they represent.

This is also true for SimMechanics, a mechanical block-diagram modelling environment for SIMULINK. However, despite the visual improvement over conventional block diagrams, the representation is still rather abstract. On the other hand, both Dymola and SimMechanics facilitate the preparation of models by offering the possibility to import components automatically from SolidWorks CAD files. Joints, masses, moments of inertia are translated but some variables, such as spring and damper constants have to be entered manually. Visualization is problematic, because the geometry information that determines the appearance of artefacts is lost [100]. The standard output of block-diagram simulations is numerical. Dymola and SimMechanics can link the output to a 3D animation of a CAD model, but the procedure is labour-intensive.

In bond graphs the connecting ports have been defined to correspond to physical connections in electrical and hydraulic systems [101]. However, in the mechanical realm they do not (*cf.* Figure 11, [96], reproduced with permission of the author), which makes the models difficult to comprehend [68]. Another drawback of these graphs is that they have been developed with discrete-component systems in mind [102]. To study behaviours of a continuum, e.g., local deformation behaviour, objects must be discretized. Yen and Masada have applied this as the 'extended bond graph method', to simulate vibration in hyperelastic thin plates. In a comparison, however, they found that the validity of results obtained with the finite-element approach (see 2.3.7) is better.

Another disadvantage is that, although bond graphs represent objects and although they can be used to model humans and artefacts together, no approach could be found in the literature to automatically derive bond-graph models from CAD models. None of the packages in the aforementioned reviews appears to offer this functionality. Thus, creating bond graph models requires additional modelling efforts. To represent a product as a bond-graph, the designer has to model it from scratch using bond graph elements. Also, the standard out-

⁹ See also: elaboration of research question (iv) on page 7.

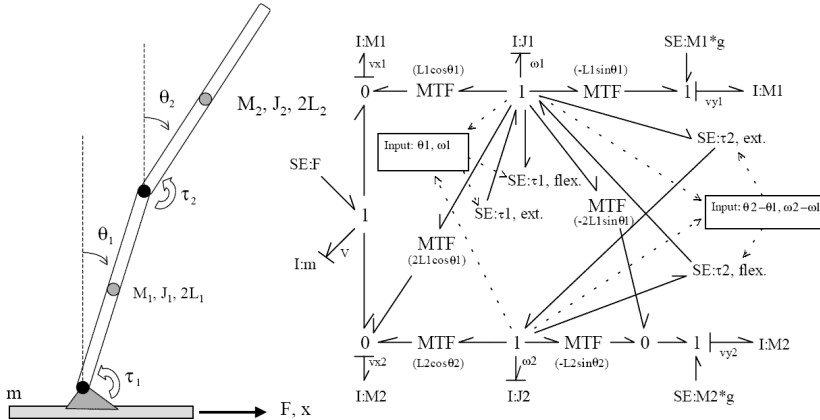


Figure 11. Schematics of the dynamics of human sagittal balance with bond graph model

put of bond-graph simulations is numerical and the graphical representation of a bond graph cannot be animated to visualize the results.

2.3.5 Simulations based on abstract entity models representing 3D schematics

Abstract entity models representing 3D schematics use graphical elements that include part of the geometry of the object. I have distinguished rigid 3D line models and skeleton-like models. Rigid 3D line models consist of connected edges (rods) and nodes arranged in 3D space. The nodes define joints and their degrees of freedom to enable kinematical simulation of linkages [103]. Usually a model-independent simulation algorithm incorporates constraint-based knowledge of kinematical laws for simulation. In most of the current applications, the functionality of nodes has been extended to make simulation of kinetostatic¹⁰ behaviour possible with springs, dampers, force actuators, etc.

Skeleton-like models represent the geometry of objects by dimensionally reducing them to forms without interior [23]. Compared to line models, skeleton-like models extend the functionality of the nodes to areas of physics outside the mechanical domain, even offering the possibility of multiphysics simulation [104]. The modelling elements contain knowledge about behavioural laws. Skeleton-like models can be considered an alternative to block diagrams and bond graphs that allows a more intuitive arrangement of mechanical components, and additionally, visualization of the main geometry. An example of commercial software for kinematical line-model simulation is Linkage Designer¹¹, an add-on to

¹⁰ Kinetostatics is kinetics without consideration of inertia effects (quasi-static interpretation of dynamics)

¹¹ linkagedesigner.com. Nowadays, most commercial software for kinematic and kinetostatic simulation uses volumetric and/or mesh-based models (see 2.3.6 and 2.3.7). Apart from line models, Linkage Designer supports simplified volumetric models. Kinetostatics cannot be simulated.

Mathematica. The software package PREDES [105] was an academic effort to enable modelling and simulation based on skeleton-like models.

To mention two applications of these models to simulate the use of artefacts, Krovi et al. [106] simulated rehabilitation aids for the disabled using kinematical and kinetostatic models, and Figure 12 shows a skeleton-like model of a hand drill created with PREDES (courtesy of I. Horváth).

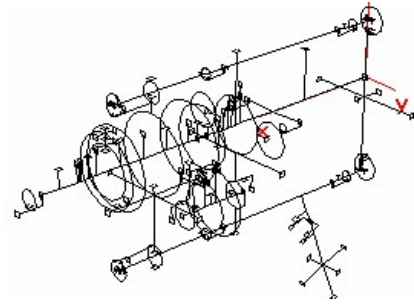


Figure 12. Skeleton-like model of a hand drill

Concerning recent application cases, 3D schematic simulation models have been more frequently applied to human simulation than to artefact simulation. In human simulations, the models are typically some kind of skeleton-like model. These have been developed for motion simulation and posture prediction [107], i.e., simulation of kinematical behaviour. The models mostly represent the human skeleton, but skeleton-like models have also been generated based on anatomical landmarks [21]. Jung and Choe [108] presented a skeleton-like model that they used to predict the reach envelope of the upper limbs. For simulations in which humans and artefacts interact the volumetric models discussed in the next two sections have been more commonly used.

Regarding the ease of preparing simulation models and interpreting simulation results, the following can be said. The available information about Linkage Designer does not mention import possibilities, but output in the form of 3D animations is possible. In PREDES, partial automatic conversion to import CAD models was possible, but conversion results were not unique. The system has not been developed to a version that can provide output in the form of animations. The applications to human modelling and simulation vary regarding their import capabilities, but the results can typically be visualized as animations.

2.3.6 Simulations based on rigid 3D volumetric models

Simulation approaches for rigid 3D volumetric models have been developed for kinematical and rigid-body kinetic behaviour. If they cover both types of behaviour, they are generally known as *multibody simulation* approaches. Purely kinematical simulation has nowadays been included in the assembly modules of most of the commercial solid-modelling CAD systems [109]. Dedicated packages such as Adams, LMS Virtual.Lab, and Simpack [110] perform multibody simulations with 3D volumetric object models. Flexible-body kinetics is limited to discrete components (springs, dampers). Knowledge about behavioural laws is not embedded in the virtual prototype but in a separate simulation algorithm. An overview of the mathematical backgrounds of the various multibody simulation algorithms can be found in [111]. Current tools seem to be in the mature stage with no significant further developments pending [112].

Multibody simulations have been applied to a wide range of consumer products, especially to transportation means such as cars, bicycles, and motorcycles [113,114], but also to domestic appliances, e.g., washing machines [115]. Figure 13 gives an example of a multibody simulation model created with Adams¹². 3D volumetric representations have been deployed in many human simulation systems, some of which are limited to kinematics simulations while others include kinetic aspects. The most advanced of these models have been based on a mix of representation forms. They will be discussed in 2.5 (more specifically, 2.5.2), because they also include aspects of control.

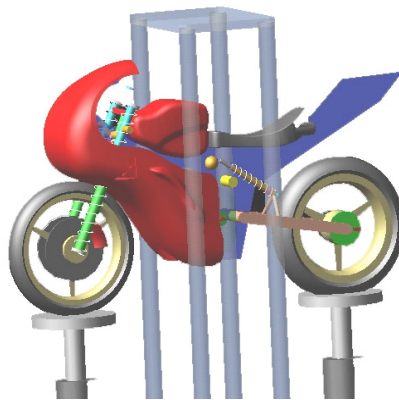


Figure 13. Multibody simulation model of a test rig for motorcycles

Most of the models that do *not* include control belong to the group of guided 3D volumetric human models, which have to be manipulated interactively. In this area, various commercial software packages have been on the market since the 1970s. They offer support for the assessment of possible postures and fields of view, e.g., SAMMIE, HECAD, COMBIMAN and CHES [116]. Apart from those, two more recently developed commercially available systems are LifeModeler¹³, which is based on Adams, and Anybody [117]. These systems simulate human motions and actuator behaviour based on user-defined (or pre-recorded) kinematical posture sequences that typically correspond to brief singular interactions in a use process, such as pulling an handsaw or hitting a golf ball [118], and not on complex sequences of interactions. However, they allow detailed investigation of individual muscle forces to optimize products for comfort during specific interactions, or to enhance human performance in sports.

An advantage of volumetric-model based simulation is that it can be seamlessly integrated with conventional solid-modelling tools for product design. The output can typically be shown as an animation of the object model. To perform ergonomic analysis with guided 3D volumetric human models, artefact models can typically be imported from CAD to investigate virtual products with virtual humans in virtual surroundings [119].

2.3.7 Simulations based on polygon mesh models

A mesh in 3D is a simplified volumetric representation created by discretizing a geometric domain into small elementary shapes. Typically these are polygons, such as tetrahedra and hexahedra [120]. The finite-element (FE) approach, which is based on simplified volumetric representations, is the oldest and perhaps the

¹² Courtesy of VI grade engineering software and services, www.vi-grade.com

¹³ lifemodeler.com

most popular form of mesh-based simulation [121]. It is based on laws that assume energy minimization in the object and on interpolation functions that are applied on the mesh. Knowledge of the behavioural laws is not included in the object representation itself but in algorithms operating on them. The FE approach was originally intended for static stress analysis [122]. Later extensions cover dynamic mechanical behaviour, heat conduction, electric and magnetic potential and hydrodynamics [123], as well as acoustics [124] and optics [125]. However, I could not find publications reporting on the use of FE-based simulation in kinematics.

Other mesh-based approaches are the boundary element (BE) and finite-volume (FV) approaches. The BE approach is based on a simplified boundary representation. Only the surfaces of objects are meshed with 2D polygons, which reduces the preparation time [126], but the simulation algorithms are more computation intensive [127]. A mesh-based modelling technique similar to the BE approach and which is used for human-body modelling in computer-graphics animation is *skinning* [128]. In this case the surface mesh is solely used to simulate shape changes during posture changes. Deformations are not calculated based on forces and energy but on surface fairing and interpolation techniques [129]. Typically the skin mesh is combined with an internal skeleton model to simulate the kinematical behaviour. The FV approach reduces volume integrals in partial differential equations to surface integrals, which is beneficial in fluid-dynamics simulation [130]. The BE and FE approaches can also be combined into an integrated approach [131].

Commercial FE software packages are, among others, MSC Nastran [132], Algor, Ansys, Visual FEA and Comsol¹⁴. Only a few commercial software packages based on the BE approach seem to be on the market, mostly focusing on one of the non-mechanical types of physical behaviour. Examples are Celsius for thermal simulation and Paragon Neo Acoustics for acoustic simulation¹⁵. Examples of commercial software using the FV approach are Micro Fluidics and STORM Solver¹⁶.

A promising advancement is the increasing support of multiphysics [133], which most of the commercial FE software packages claim to offer. Multiphysics simulations are typically performed as multiple (typically two) simulations running in parallel, each covering a distinct field of physics. They exchange data after each time step. Cross et al. reviewed the state of the art in multiphysics simulation based on polygon mesh methods [134]. They distinguished simulations with *loosely coupled* and *closely coupled* data exchange. In the most loosely coupled simulations there is a one-way exchange of data. An example is a co-simulation involving heat flow and mechanical deformations in an object, where the thermal expansion due to changes in temperature is included in the mechanics simulation, but geometric changes resulting from deformation are not considered in

¹⁴ mscsoftware.com, algor.com, ansys.com, comsol.com.

¹⁵ integratedsoft.com, paragon-neo.com

¹⁶ www.phoenixbv.com, www.adaptive-research.com/storm_solver.htm

the thermal simulation. In the most tightly coupled simulations all the changes in each parallel simulation are considered in all the other simulations.

Cross et al. concluded that among the available systems loosely-coupled multiphysics simulation involving two phenomena is nowadays widely supported, but tightly coupled multiphysics remains a computational challenge. Presented examples are typically limited to simple geometries (2D or rotationally symmetrical), where simulations of processes with a duration of a few seconds require several hours of simulation on high-performance parallel computing systems [135]. Figure 14 shows a simulation of mechanical deformation, electric current and heat flow with Ansys Multiphysics (reproduced with permission of Ansys, Inc.). The animation frames show the changes in shape and temperature distribution. Note that the simulated model is essentially 2D.

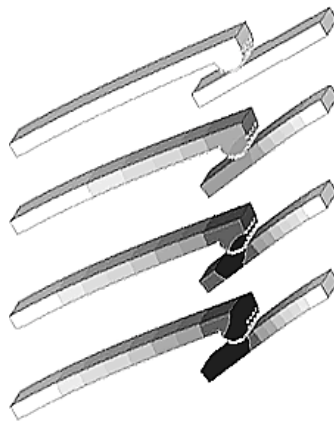


Figure 14. Multiphysics simulation of a switch.

It has to be mentioned that loose coupling can also be used to perform co-simulations of polygon mesh models and multibody dynamics [e.g., 136]. Most of the aforementioned vendors of multibody simulation packages (in particular, Adams and UMS Virtual.Lab) offer add-ons that allow export of forces to a FE module, which computes stresses and deformations, which then can be displayed in the animated multibody simulations. However, the changes in geometry as a result of deformation are not considered in the multibody simulation.

In simulations of product behaviours the FE approach has frequently been applied. Friswell et al. [137] simulated vibrations in golf clubs. Middendorf [138] investigated mechanical stress in a motorcycle suspension fork and magnetic fields in the rotor of an electric motor. Figure 15 shows two more examples from commercial practice: deformations in a ball and bat, and stress distribution in a lounge chair¹⁷. Hanna [139] presented examples of FV-based airflow simulations around a ski jumper, a racing-car, and a Frisbee using commercial software.

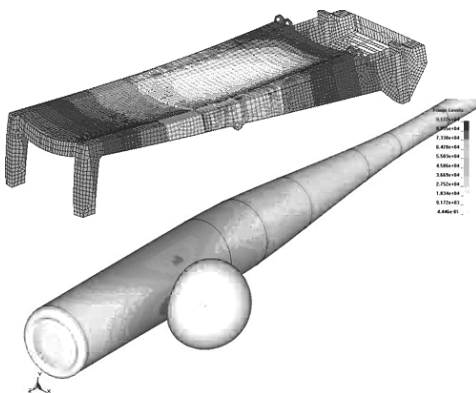


Figure 15. FE-based simulation of consumer durables.

¹⁷ Baseball bat courtesy of predictive engineering, (predictiveengineering.com); lounge chair courtesy of BPO, (bpo.nl)

Simulations based on polygon meshes have been frequently applied in human simulations. In biomechanics, where FE-based simulations have been applied from the second half of the 1970s, *static* and *dynamic* application of FE models can be distinguished. In the literature, several publications focus on modelling the complex material properties of different tissue types. Based on measurements from human body samples, approximations of the elastic and viscoelastic deformation behaviour as first-order, second-order or even third-order have been put forward [e.g. 140].

Chow and Odell [141] applied a static FE model with bilinear elastic tissue properties to predict body deformations of sitting persons. Cheung et al. [142] developed a static model for the prediction of stress in a foot during standing. They assumed the bones and ligaments to be linearly elastic and the soft tissues to be hyperelastic. Bandak et al. [143] applied a dynamic finite-element calculation to study the deformational effects of axial impact on the human foot, in which the deformation of bones is modelled as linear viscoelastic.

Koch et al. [144] included actuator behaviour by applying an equation-based model of muscle contraction in their dynamical FE model of facial-expression forming. In this model, tissue deformation is considered linear elastic. A non-linear FE model of muscle contraction and passive muscle stretching using a NURBS-based geometric representation is presented in [145]. Sun et al. [146] simulate perception behaviour with an FE model of the human ear, assuming linear viscoelastic behaviour of the cochlear fluid and linear elastic behaviour of the other ear components.

The BE method in applications to human simulation has typically been used for simulations of non-mechanical physical behaviour, as has been the case with artefact applications. Thiebaut and Lemonnier [147] presented an application to human heat transfer, and Bradley and Pullan [148] used the approach to simulate human electrical-potential behaviour. Figure 16 shows an example of skinning from the entertainment industry¹⁸.

To simulate kinematics together with skinning, Seo and Magnenat-Thalmann [149] developed a skin-mesh model that was combined with volumetric internal elements. The only examples I could find in the literature where mesh-

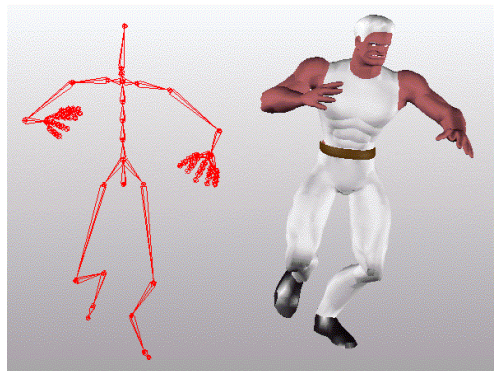


Figure 16. Combination of a mesh-based deformable skin model with a kinematical skeleton-like model in a computer game

¹⁸ Image courtesy of Okino Computer Graphics, okino.com/conv/skinning.htm; 3D model: ©Autodesk, Inc.

based models have been applied to simulate artefacts together with humans involve prostheses [e.g., 150].

Polygon mesh-based simulations of physical behaviour can be performed based on shape models created with CAD systems, but these must be pre-processed using a meshing algorithm, which is typically included in the simulation software. Modifications have to be performed on the CAD model, which must then be re-meshed for each simulation run. All the abovementioned commercial tools offer output in the form of animations. Usually, different ranges of investigated variables correspond to differently coloured zones. During the animation, the distribution of the coloured zones over the artefact change dynamically according to the computation results. Motion and mechanical deformation are also visualized directly in the 3D object representation.

2.3.8 Simulations based on particle-based and meshless models

Discretization of particle-based and meshless models is not based on polygons, but either on (i) a mesh of dimensionless particles populating the geometric domain or (ii) subdivision of the functional space underlying the geometric domain. The particles in particle-based simulation are typically connected by springs and dampers (in solid mechanics) or by implicit surfaces (in fluid mechanics). In solid mechanics, the approach has been applied to simulations of deformable objects, including viscoelasticity, plasticity, and fracture [151], where its advantage lies in producing realistic animations in real time. Mechanical applications included collision of deformable bodies [152], anisotropic material behaviour [153] and rigid-body dynamics [154].

The fluid-mechanics applications I found in the literature have been implemented for the entertainment industry, where realistic visual appearance is more important than validity of the outcomes [155]. In that application area, the main challenge is to generate a visually realistic surface model [156]. A ‘low-threshold’ approach that can be considered a variation on particle-based simulation is *discrete flexibility simulation*, which enables simulations of large mechanical deformations by applying a low-resolution discretization to 3D volumetric bodies in conventional multibody simulation software, and connecting these by spring-dampers [157]. The ‘particles’ in discrete flexibility are typically not dimensionless. Meshless approaches are based on either meshfree Galerkin methods or on Rvachev’s function method. Details on these methods can be found in [158] and [159].

Most software for meshless and particle-based engineering simulation is still in the stage of academic research and development. Apart from applying it using conventional multibody systems (discrete flexibility), the only exception so far has been the commercial software package FieldMagic¹⁹, which has been developed based on Rvachev’s function method and which does not seem to be widely used yet. No examples could be found of application to products by the industry.

¹⁹sal-cnc.me.wisc.edu/Research/meshless/meshfree.php

In human simulation, particle-based models have typically been used to model and simulate mechanical behaviour based on mass-spring models with the objective to create animations in computer-graphics applications. Zhang et al. [160] present a three-layer mass-spring model representing the epidermal, dermal and hypodermal layers of the facial skin for dynamic simulation of deformations. Nedel and Thalmann [161] applied mass-spring particle modelling to simulate muscle contraction in a model of the human body. Actually this model was based on mixed representations, also including skinning and skeleton elements. Their inverse dynamics-based simulation, which required posture changes that had been pre-defined in animation frames as input, covered kinematics and rigid-body kinetics of the skeleton, deformations and actuator behaviour of the muscles, and deformation of the skin based on skinning. Mechanical deformation of soft tissues other than muscles was not taken into account. Similar mass-spring models, mostly applied to the musculature of animals and artificial life forms, have been discussed in a survey by Cerezo et al. [162].

Particle-based and meshless approaches have several advantages over polygon mesh-based approaches [158,159,163]. First, extreme deformation and even fracture of objects can be simulated without the need for re-discretization on the fly. Secondly, the number of computation-intensive preparation steps to create simulation models from CAD is reduced. This is also an advantage if point cloud models are used that have been obtained from 3D-scans of existing objects [164].

The third advantage is the possibility these methods offer to study intermediate changes *during* simulation time steps. Most polygon mesh-based simulations only compute the minimum-energy state at the end of each simulation time step, thereby ignoring intermediate effects such as oscillations. A further advantage of meshless approaches is that they offer the possibility of non-mechanical, even closely coupled multiphysics-type of simulations. From the available examples it appears that, similar to the state of affairs in polygon mesh-based approaches, multiphysics simulations are still limited to the investigation of two concurrent phenomena in a system that is typically reduced to a simple 2D model.

Disadvantages of meshless compared to polygon mesh-based approaches are that simulation is more computation-intensive and that it is more difficult to define boundary conditions in the model [165]. Regarding the ease of interpretation of simulation results, it can be said that all the investigated methods and examples offer animations similar to the ones resulting from the polygon mesh-based approaches.

2.3.9 Discussion of the findings on behavioural simulation approaches

In the preceding sections a variety of behavioural simulation approaches has been discussed. To enable comparison according to the criteria from 2.2.3 that have been selected in 2.2.6, an overview is given in Table 1. Related to the criteria, the findings can be summarized as follows:

Table 1. Comparison of behavioural simulation approaches

<i>Simulation approaches (by type of construct on which the approach is based)</i>	<i>Criteria</i>										<i>ease of preparation (a)</i>	<i>ease of interpretation (b)</i>	
	Range of behaviours covered												
	rigid-body kinetics	deformations	kinematics	fluid mechanics	acoustics	optics	thermodynamics	electricity	magnetism	multiphysics support			
Algebraic descriptions											±*h	–	–
Quantitative algorithms											±*h	–	–
Qualitative algorithms											±*h	–	–
Conventional block diagrams												–	–
Dymola, SimMechanics												±	–
Bond graphs												–	–
3D line models												±	+
Skeleton-like models												+	–
Rigid 3D volumetric models												+	+
Polygon Mesh-based models												+	+
Multibody + polygon mesh												+	+
Particle-based models												+	+
Discrete flexibility												+	+
Meshless models												+	+

supported




supported with limitations (see notes)

unsupported

+ high

± medium

– low

 supported
 supported with limitations (see notes)
 unsupported

+ high
 ± medium
 – low

Notes:

- *a (–) very limited or no automated conversion from CAD models
 (±) automated conversion of specific variables from CAD models
 (+) automated conversion of geometry from CAD models
- *b (–) no animation of object model provided (or only possible by manual linking)
 (+) animation of object model provided
- *c modelling limited to systems built up from discrete components
- *d modelling and simulation of non-mechanical behaviour supported only by Dymola (in SimMechanics, conventional block-diagram elements need to be added).
- *e static behaviour not supported
- *f might be theoretically possible, no software available
- *g loosely coupled: geometric changes ignored in kinematics simulation
- *h complexity of equations/algorithms is limited from a practical viewpoint
- *i deformation simulation limited to low-resolution particle systems
- *j limited to simple and/or loosely coupled models due to computational complexity

- *Range of behaviours covered.* Although two distinct typifications of behaviours have been put forward in 2.2.4 for humans and in 2.2.5 for artefacts, it turned out that the differentiation in range of covered behaviours can be expressed almost completely by referring to the areas of physics listed in Figure 7, which applies to both artefacts and humans. Of the ‘specific’ human behaviours in Figure 8, *actuator behaviour (force exertion)*, *kinematic behaviour*, and *kinematical behaviour* can all be mapped to fields of mechanics in Figure 7, and *perceptual behaviour*, *cognitive behaviour*, and *control behaviour* are all related to the control loop, for which simulation ap-

proaches will be evaluated in 2.4 and 2.5. The only exceptions are that in 2.3.2, 2.3.6, and 2.3.7 some specific behaviours related to vision, actuation, and hearing have been mentioned that could be modelled and simulated with algebraic descriptions, 3D volumetric, and 3D mesh-based human models. At a closer look these behaviours are actually based on physical phenomena that are also included in Figure 7. Therefore, to address ranges of behaviours in Table 1 I have only listed the coverage of the areas of physics for each simulation approach.

Algebraic descriptions and algorithms (both quantitative and qualitative) appear to be the most universally applicable. Two-dimensional graphical object representations are also versatile, but limited to systems built up from discrete components. In the group of 3D representations, the skeleton-based approach, which does not seem to be adopted by commercial software, appears to offer the widest (potential) coverage of behaviours. Most of the discretized-model based approaches lack support of kinematics, and they offer multiphysics support only for simple models. The other approaches based on 3D models hardly support modelling and simulation outside the field of (rigid-body) mechanics.

- *Relevance of the scope.* In particular, the approaches based on graphical entity models (block diagrams and bond graphs) focus on simulating the behaviour of systems built up from discrete components, e.g., in mechatronics. Simulating physical effects in continua, which is often important in the use of consumer durables (for example, deformations in human tissues or in seats), is difficult using these approaches.
- *Ease of preparation.* According to Table 1, the approaches based on 3D object representations require the least preparation effort because they mostly offer automatic conversion from CAD models. A disadvantage is still that non-geometry related physical properties (e.g., thermal conductivity, or Young's modulus) are absent in CAD models and have to be entered during the simulation preparation. Also, the automated pre-processing needed for the discretized model-based approaches (e.g., meshing and remeshing) is computation-intensive and thus time consuming.
- *Ease of interpretation.* The overview in Table 1 shows that the various approaches based on 3D object representations can directly produce animations of simulated behaviour. The output of most of the other approaches can also be connected to 3D representations to provide animations, but this requires manual efforts.

Interpreting these evaluation results, it can be said that among the current behavioural simulation approaches the combinations of multibody simulation with polygon mesh-based models appear to offer product designers the most advantages. Their most important drawbacks are that (i) rigid-body mechanics and deformation simulations are only loosely coupled, and (ii) the possibility to include other areas of physics is also limited because of computational complexity. The first disadvantage is especially important in human-product interaction, where large deformations play an important role in the interactions [50], e.g., in

grasping and in interaction between human buttocks and seats. If we want to simulate these interactions, close coupling is needed to include geometric changes due to deformations in kinematical simulation. As possible new simulation paradigms, particle-based and meshless approaches seem to emerge, but commercially available tools are still scarce and, apart from applications to entertainment and gaming, the approaches have not been adopted by the industry.

2.4 Introduction to the review of control in simulations

2.4.1 Scope of the review

This subchapter and the next one cover both ‘the control of simulations’ and ‘simulation of control’. Although fundamentally different notions, they are often implemented based on the same principles. The distinction between ‘control of simulation’ and ‘simulation of control’ refers to the system boundary considered for simulation.

Control of simulations implicates that a simulation receives inputs from (and, in the case of a closed loop, produces outputs to) a control mechanism that is considered to be external. In the case of software-in-the-loop (SIL) simulation, this external control mechanism has been programmed into a piece of software connected to the simulation software [39,40]. The term ‘control of simulation’ especially applies if processing of control cannot (or not completely) be considered a simulation itself, because it is not based on models of behaviour. This is for instance the case if the control has been specified as instructions or as a conjectured set of actions (e.g., a set of scenarios).

On the other hand, the term simulation of control applies in all cases where the execution of control is simulated [166], i.e., it is processed based on behavioural models. In that case it is not important whether the control is considered to be inside or outside the system boundary of the simulated system.

An important fundamental distinction in both cases is that between continuous control and logical (discrete) control. Continuous control mechanisms are based on signals that directly correspond to physical phenomena as listed in Figure 7. The behaviour of such mechanisms is typically simulated based on one of the simulation constructs already evaluated in 2.3. For the review in 2.5 it is therefore sufficient to discuss the application of continuous control simulations to humans and artefacts. In this case, separate consideration of the concept ‘control of simulation’ is irrelevant, because *continuous* SIL control is always based on simulation with behavioural models [e.g., 167].

Logical or discrete control mechanisms need a different treatment, since they are either *simulated* or *executed* based on constructs that have not yet been introduced in this chapter. In these mechanisms, such as (in the case of artefact control) microcontrollers and programmable logic devices (PLDs), logical operations are performed on information that is *interpreted* from physical effects that can be observed in signals. For instance if the voltage V produced by a device represents ‘0’ if $V < 0.5V$ and ‘1’ if $V > 1V$, then the output ‘100111’ is the ab-

straction or interpretation of a continuous change in output voltages during a certain interval of time (Figure 9).

Constructs operating on such interpretations disregard the physical background of the signals. Also, they can be based on other foundations than modelling of behaviour, which makes the distinction between 'control of simulation' and 'simulation of control' relevant. In 2.5.2, the various representations for logical control are reviewed based on their representation potential and ease of use.

In the next two sections (which correspond to 2.2.4 and 2.2.5 in the discussion of behavioural simulations) the specific types of control in artefacts and humans are introduced. In the case of humans, the stages of interaction control that were already mentioned in 2.2.5 are further discussed.

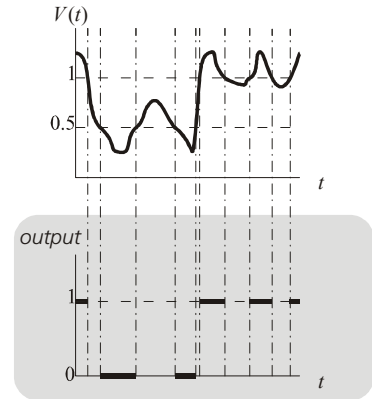


Figure 17. Discrete interpretation as output of a continuous change in voltage

2.4.2 Control behaviours of artefacts

Mechanisms for artefactual control mechanisms are subdivided into two main categories, namely, analogue control and digital control. These correspond to 'continuous' and 'discrete' control, respectively. In continuous control, a control signal constantly applies corrections to a controlled signal so that the corrected output signal remains close to a set value. A common control mechanism is the closed feedback loop where the control signal is derived from (a measurement of) the output signal, and which is often based on the proportional/integral/derivative (PID) algorithm [38, p. 610].

Continuous control in artefacts is usually realized by hardware components that process physical quantities (e.g., voltage or pressure) as signals. Analogue electronics is most commonly used, but continuous control can also be hydraulic or mechanical (e.g., centrifugal governors) [168]. Analogue control mechanisms are typically simulated based on differential equations. These equations either regard the control signals according to their physical nature as electrical voltage, hydraulic pressure, etc, or as a continuously changing flow of information derived from it [169]. On the other hand, embedded digital control mechanisms, such as microcontrollers and programmable logic devices (PLDs), are usually simulated as discrete-event systems performing logical operations on information that has been interpreted from signals [170].

2.4.3 Specific human behaviours related to control of interactions

Theories on human interaction behaviours cover a wide variety of responses to external stimuli, such as walking, looking, reaching, grasping, drawing, key-boarding, speaking, etc. [171]. Human motor scientists have developed various

approaches to control modelling. The literature agrees that signals from the sense organs are processed by the central nervous system and alternately by the brain. This happens through a series of stages, which allows a more detailed segmentation of some of the main behaviours in Figure 8. Several interpretations have been proposed for this structuring.

For instance, Stelmach proposed a decomposition scheme with five generic stages (Figure 18), namely, detection, recognition, decision-making, response selection and response execution [172]. These generic stages have a lot in common with other subdivisions, such as the ones proposed by Wickens and Hollands [173] and by McRuer [174]. In fact, detection and recognition are considered *preparations* for the inputs of control, while decision-making, response selection and response execution represent the actual control. Involving conscious cognitive processes, decision-making is typically considered ‘high-level’ control, whereas response execution (i.e., the actual motion of limbs) is treated as ‘low-level’ control [175].

Decision-making in human motion control is not to be confused with other high-level decision-making actions on a social level [176]. A common approach in contemporary cognitive psychology is to model decision-making based on logic [177]. Response selection has been considered a hierarchical structure of options in which a choice is made based on the task at hand [175]. Response execution has been described as an eye-limb coordination exercise [178]. Findings over the past decennia indicate that the human brain specifies the input to response execution as movements based on positions, angles, velocities, and angular velocities rather than on forces and accelerations [e.g., 179]. The theory of invariants, i.e., generalized and parameterized motion patterns, allows us to ignore the influence of eye-limb coordination in response-execution modelling [180].

As was just mentioned, modelling of decision-making is typically done based on logic. Regarding the other stages of control, there is still much debate in the literature whether these are to be modelled as discrete (logic-based) or continuous²⁰. The *discrete-information processing the-*

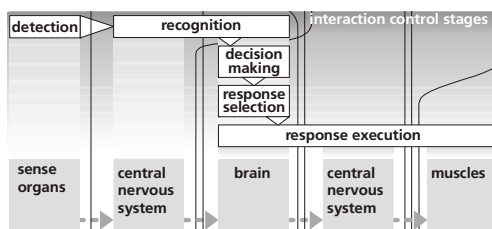


Figure 18. Generic stages of human control in interaction inserted into the diagram of Figure 8.

²⁰ This is reflected in the two theories that have been used to represent control behaviour of humans, namely, the discrete-information processing theory and the continuous information-processing theory. Both theories can be considered elaborations of the theory of human information-processing [181], which initially built its foundations on discrete information-processing models in the 1970s, with continuous-processing based alternatives becoming dominant from the second half of the 1980s [182].

ory²¹ considers and models the human as a processor of information, comparable to a computer [172], and as a composition of receptors, effectors, and an intervening control system, with information processing concerned primarily with the operations of the control system. The *theory of continuous information processing* is also known as the theory of proportional control [183].

The applications of this theory in the literature concentrate on response execution. Costello [184] reserved continuous control models for modelling small corrections, while he considered large-scale movements to be represented by discrete models. Sanders [185] proposed two stages of response execution, namely, (i) response programming, which suggests logical control, and which seems to correspond to Costello's large-scale movement planning, and (ii) motor adjustment, which is supposed to produce "instructed muscle tension", in other words, it specifies the force required. This also seems to be in agreement with findings that the velocity and position control signals that are input to response execution are generally considered to be discrete and pulsatile [11].

Concludingly, findings from the literature (which does not appear to be decisive) suggest that computational modelling and specification control of human movements can be kept relatively simple by applying two principles. The first is to implement generalized and parameterized motion patterns. If these *invariants* are available, they can be used to circumvent simulation of eye-limb coordination, and thus disregard the role of perception in motor adjustments. The second principle is to model/specify response selection and execution as logical control, with the exception of the final translation of motion patterns to muscle forces, which is modelled as continuous (e.g., PID) control.

2.5 Review of control of simulations and the application of scenarios to simulation

2.5.1 Simulation of continuous control behaviour of artefacts and humans

As was explained in 2.4, the constructs that are used for simulation of continuous control have already been discussed in 2.3. Since control behaviour is often unrelated to geometry and shape, the typical simulation constructs for continuous control behaviour are the non-spatial ones that have been reviewed in 2.3.1-2.3.4. Consequentially, the same software packages can also be used for modelling and simulation.

Several examples of simulation models of continuous control based on algebraic and calculus-based representations, algorithms, block diagrams, and bond graphs can be found in [169] and many other publications. Since many commercial packages for continuous simulation (e.g., finite element analysis, multibody

²¹ Also referred to as the 'black box theory', or, confusingly, as the information-processing theory.

dynamics) offer an interface with it, Matlab Simulink is often used to simulate control based on block diagrams [186,187].

Continuous simulation of human control behaviour is typically applied to investigate response execution of movements. In the investigated literature three types of models have been used: (i) conventional (PID) control loops, (ii) pre-calculated posture frames based on optimization algorithms, and (iii) artificial neural nets (ANNS). Examples of behaviour simulated as a conventional control loop are eye-hand coordination in vehicle control [188] and interactive compensation of machine behaviour by McRuer [174].

In McRuer's loop, the machine reacts to human input and to external disturbances that the human needs to compensate. In both simulations, the physical transfer of information entered by the human into the machine – e.g., by operating an interface element – is a step that is skipped. An example of a simulation in which the physical effects of control are included, but no interaction with artefacts, was proposed by Multon et al. [189]. In their dynamical model of the human arm, simulation of conventional control behaviour was combined with actuator behaviour, kinematical behaviour and rigid-body kinetic behaviour.

The second approach to simulate continuous human motion control is to calculate incremental changes between specified start and end postures by applying optimization algorithms. It can be seen as an extension of the direct transfer of motion-capture data to simulation control, which is the standard form of control offered by, for instance, the LifeModeler human simulation package [e.g., 190]. This approach is inflexible because it only allows simulations of motions exactly as they have been captured from a specific human subject.

To overcome this disadvantage, Park et al. [191] used a database with so-called root motions captured from human subjects between a limited set of pairs of start postures and end postures. To find motions between other pairs of postures, the most similar root motion was retrieved from the database and then adapted to match the new start and end posture by minimizing the deviation from the root motion. For the Santos virtual human, Yang et al. [192] implemented a different form of optimization. They used a multi-objective optimization algorithm to calculate the most likely intermediate postures based on minimum potential energy, preference for 'neutral postures', and three other factors.

One disadvantage of these approaches is that the exact end posture (and not only the position of a landmark, such as the fingertip) must be specified beforehand. Another potential disadvantage of pre-calculating motion frames is uneven distribution of computational load over the simulation runtime, because all postures must be computed before the simulated movement can start. Nevertheless, it has been claimed that Santos can perform simulations in real time [193].

The third group of approaches is based on ANNS, which form a particular class of block diagrams [89]. They can be considered discretized models of hu-

man body components, that is, of biological neural nets²². ANNs differ from conventional simulation algorithms because of their ability to learn and self-organize, to generalize from training data, and to process information in other ways normally thought of as intelligent [194].

An example is the simulation of steering behaviour of airplane pilots that was performed by Martens [195]. In this simulation the actuation of muscles and the physical interaction between the human

and the controls of the plane were not considered. Instead, the output control signals of the human were directly converted to positions of the control stick. Kim and Hemami [196] used an algorithm combining an ANN with nonlinear equation-solving to simulate motion control, kinematical and rigid-body kinetic behaviour of the head and torso. The ANN acted on the output of a 'desired trajectory generator', which represented cognitive decisions about movements of the head. This unit was left out of the simulation.

Reil and Husbands [197] combined ANNs with genetic algorithms in a simulation approach for human walking based on evolutionary selection principles culminating into a best-performing network. The approach has been further developed to the commercial software package Endorphin (Figure 19, reproduced with permission of NaturalMotion Ltd.²³) which has successfully been applied to generate human motion animations for the entertainment industry [198]. To prepare simulations of action sequences, typical 'behaviours' selected from a library (e.g., 'jump', 'stagger', 'writhe') are scheduled on a timeline together with events such as predefined occurrences of forces acting on the human body [199].

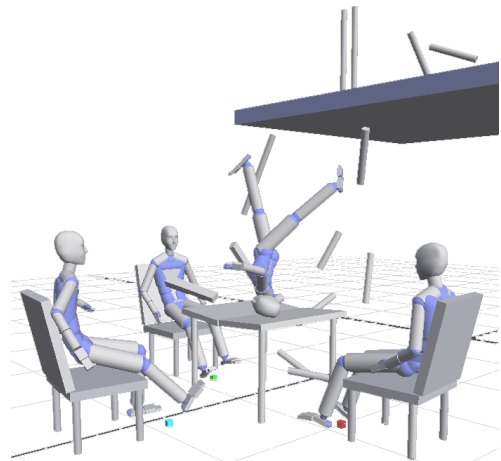


Figure 19. Simulation of a human falling from a balcony, created with Endorphin

2.5.2 Logical control: formal representations

Logics-based representations of control mechanisms and processes are typically finite automata or state-machine representations, which prescribe transitions between states of a system. A transition between states occurs if the automaton receives specified input [200]. Output can also be assigned to transitions or

²² However, usually there is no one-to-one correspondence between the components in an ANN and the components of a biological neural net.

²³ naturalmotion.com

states [201]. Three categories of logics-based representations can be distinguished: (i) formal language-based (using procedural logic in the form of text and declarative codes), (ii) algebraic/numeric (using matrix representations, Boolean algebra, and temporal logic), and (iii) graphical representations [145].

Graphical representations include informal graphical symbol constructs typically based on directed graphs [202], and formal, numerically processable symbolic constructs. I have concentrated on these below. Graphical representations are generally easier to comprehend since (i) hierarchy and parallelism in complex systems can be shown more clearly, (ii), graphics allow selective reading depending on the level of details needed, and (iii) the number of concepts to be held in short-term memory is smaller [203]. Graphical representations with multiple uses are state transition diagrams (STDs) [204] also known as event graphs [205], Markov models [206], Petri nets [207], and statecharts [208]. Additionally, a large number of ‘dialects’ of varying significance has been developed to extend these representations, most notably Petri nets. The most important dialects of Petri nets are stochastic, timed, and high-level Petri nets [209], and the most important dialects of statecharts are stochastic and timed statecharts, as well as modecharts [210-212]. Various software packages exist for specification, modelling and simulation of STDs [213], Petri nets²⁴ and statecharts [214].

Considering its representation potential, the STD has become the basic representation for finite automata. It specifies transitions between global states of the whole control system and it can be converted to any other of the above ‘advanced’ representations. Other conversions are also possible, e.g., from Petri nets to statecharts [215,216] and vice versa [217]. It should be noted that these conversions have been elaborated only for Petri nets and statecharts with particular characteristics.

The ‘advanced’ representations have been developed to support (i) probabilistic transitions (Markov models, stochastic Petri nets, and stochastic statecharts), (ii) timing, i.e., delayed transitions (timed Petri nets, timed statecharts, and modecharts), (iii) distributed or concurrent states (Petri nets and statecharts), and (iv) hierarchical decomposition (statecharts and high-level Petri nets). Probabilistic transitions are needed to model non-deterministic systems.

The capability to represent timing makes it possible to model countdown timers and latency in the control system. Distributed states, concurrency, and hierarchy are applied to avoid ‘state explosion’, i.e., the need for a large number of states and transitions in STDs and Markov models, which makes these representations hard to work with [212]. Based on these considerations, it can be concluded that statecharts and Petri nets, which support distributed states, concurrency, and hierarchy (either by default or by using dialects), offer the highest representation potential.

²⁴ informatik.uni-hamburg.de/TGI/PetriNets

2.5.3 Logical control of artefact simulation and simulation of logical control in artefacts.

In physical artefacts, digital circuits and embedded software are the typical discrete subsystems for which automata have been used as a representation. The input corresponds to signals that these subsystems receive from sensor components and the output to signals that they transmit to actuator components [218]. In the development of products and systems, automata are often deployed as virtual prototypes of digital circuits or embedded software.

Letting automata (which may be linked to a simulation model) execute the instructions intended for the physical discrete subsystems is a form of *emulation*, which is a specific type of simulation. Emulation has been defined as “software added to one computer system to enable it to execute programs written for another system” [219]. It is based on an *exact translation of instructions* rather than on a model of behaviour. If emulation is successful, it is possible to automatically create hardware designs and even fully functional (embedded) software from automata representations [220,221].

Specifically in simulations of manufacturing logistics, Petri nets have become the preferred virtual prototyping representation of the control of machines and plants [212]. In a Petri net, distributed states are modelled with discrete tokens. They can be considered to represent processed units as they are distributed in the plant [e.g., 222]. The popularity of Petri nets in industrial automation can also be attributed to the fact that the international standard for sequential function charts, which are used to design PLDs, is based on them [223].

Statecharts have become the prevalent representations in virtual prototyping of most other products and systems. Vahid and Givargis [170, p.217] elaborated on the use of statecharts in virtual prototyping of elevator control systems. Kendall and Jones [214] used statecharts to control simulations (and afterwards physical prototypes) of door-locking systems and other subsystems in cars. Praehofer and Pree [224] used statecharts to simulate the behaviour of an electric kettle: discrete simulation of switching of the thermostat and the level sensor was performed concurrently with continuous simulation of temperature and level variations of the water.

Other automata representations have also been applied in simulations of consumer products. For instance, Martell [225] used an STD to emulate the user interface of a microwave oven in human-in-the-loop simulations, and Christensen [226] used a Petri net to emulate a linking device for networked audiovisual equipment.

2.5.4 Logical control of human interaction simulation and simulation of logical control in humans.

Human control of interactions has been subdivided into the stages of decision-making, response selection, and response execution (see Figure 18). In the discussion below I have concentrated on models and specifications that have been simplified so that perception did not have to be considered.

Decision-making has been modelled based on procedural logic, in systems such as AM, EURISKO, SOAR, ACT-R, and EPIC [227-230]. The logic is represented in these ‘cognitive architectures’ as production rules, in accordance with common practice in cognitive psychology [177].

Response selection as a hierarchical structure in accordance with [175] has for instance been proposed for classification of grasping responses by Cutkosky and Howe [231]. This scheme sorts the relevant grasping responses based on the shape and dimensions of the object. Response-execution models that allow us to disregard perception have been proposed based on invariants (i.e., generalized and parameterized motion patterns) [180]. Bullock and Grossberg [232] used invariants identified in [233] to describe human reaching with differential equations, for which logic-based algorithms provided parameter values. There are indications in the literature that invariants also apply to grasping [234]. However, due to the large number of possible variables and the redundancy of applicable grasping patterns, this issue needs further study [235].

A radical approach to modelling of interaction control is to bypass motion control of human limbs, and directly model the effects of human control on the artefact with which the human interacts. This is particularly feasible if the artefact is a given specific product, e.g., an aeroplane or a car. For instance, using an STD combined with evolutionary algorithms, Fogel [236] modelled response execution of pilots as motions of the steering components of the aeroplane rather than motions of the arms and hands. Liu and Salvucci [237] presented a model of human decision-making based on a hidden Markov chain, which was used to simulate driving behaviour by computing accelerations and direction changes of cars. In 2.5.1, the same ‘radical’ approach was also observed in continuous control simulations [174,195].

In a small number of other, more advanced approaches, the models have also been built with the intention to include physical human control of interaction with hardware. These combined human models also incorporate various subsets of constructs and behaviours that have been reviewed in 2.3. Perhaps the most well-known example is the *Jack* manikin that has been developed at the University of Pennsylvania [238,239] and that is nowadays commercially marketed by Siemens PLM as *Vis Jack*²⁵.

Jack is a semi-autonomous virtual human that makes decisions based on commands interactively entered by the user. These commands are specified in near-natural language, and they involve the highest level of decision making, e.g., ‘walk around the room’ [240]. They activate behavioural models of lower-level decision-making that have been coded in a language-based (C++) finite automata representation called parallel transition networks (PaT nets). Lower-level control is based on stored sequences of animation frames. Additionally Jack’s simulation algorithms cover actuator behaviour, rigid-body kinetic behaviour and kinematical behaviour. Artefacts are imported from a CAD environment

²⁵ plm.automation.siemens.com/en_us/products/tecnomatix/assembly_planning/jack/vis_jack.shtml

into JACK's surroundings, where they can be moved by the manikin. Weights are assigned to objects for kinetostatic simulation of loads on JACK's joints.

Based on ACT-R, Carruth et al. presented an extension to the Santos virtual human (see 2.5.1) to simulate cognition and vision [241]. Using this ACT-R/DHM extension, human-product interaction tasks could be simulated as follows. So far, the vision simulation in ACT-R/DHM has been the furthest developed part. It perceives and recognizes interaction features (e.g., buttons) in a 3D static product model created with DSS Virtools [242]. Based on the perceived features, the cognition simulation of the cognitive architecture synthesizes motor control commands to interact with the product (e.g., reach for a button), which are transferred to Santos. Santos then generates the actual movements based on its low-level control simulation algorithms. The control loop is closed by the vision simulation that 'perceives' the movements of the 3D human model with respect to the commanded interaction with the 3D product model, and feeds the progress of the task back to the cognition simulation.

So far, the project does not include behaviours manifested by artefacts, and it is limited to control of simple tasks. A disadvantage of using cognitive architectures is that new tasks must be included by adding modules to the production system, which is a labour-intensive endeavour that involves formulating text-based rules using procedural language.

As an alternative to control exerted by simulated cognition, the original developers of Santos have proposed the consideration of *scenarios* to schedule high-level motor control in human-product interactions [243]. So far, the concept of scenarios has been widely applied as a simulation approach in software engineering, but its application to forms of interaction other than human-computer interaction has largely been limited to the use of informal representations to support creative processes [4-6].

Since the concept of scenarios has been hypothesized as a carrier for scheduling and organizing multiple consecutive interactions for the research reported in this thesis, it has been further elaborated in the next subchapter by (i) giving an account of the literature explaining the concept and (ii) reviewing its current applications to simulation of use processes.

2.5.5 Scenario-based control in interaction simulation

A scenario has been defined as a possible way for a human user to control his interaction with a given product in given surroundings. Its execution usually means going through a series of choices from subsequently available options. In the context of use of consumer products, Stanton and Baber [244] explained scenarios by referring to Newell and Simon's theory of human problem solving [245]. They adopted the viewpoint that the goal of using a product is to solve a problem. In practice, the user moves through a decision tree from the initial state 'problem unsolved' to the goal state 'problem solved', selecting between available operations. Each junction in the tree has various paths representing state-transforming operations, of which one is selected. Each of the possible routes that connect junctions is a scenario of use, and the set of all possible scenarios

forms a scenario tree [246].

This common interpretation has been criticized for two reasons. Firstly, if use actions fail, the scenario does not end in a goal state 'problem solved'. Therefore, 'negative' scenarios should also be considered [247]. Secondly, the tree is known to have limitations in terms of flexibility of representation. Therefore, more general terms like 'organized sets of scenarios' or 'scenario networks' have been suggested to include other, more flexible arrangements of possible scenarios [248,249].

There have been efforts to use organized sets of scenarios to achieve logical control over concrete operation and/or simulation processes. These have been typically combined into one control model by using one of the representations discussed in 2.5.2. Organized sets of scenarios have become common in software engineering, where they are used for requirements specification, verification, and prototyping [248,249]. Since software prototyping is usually done by executing or emulating the program under development, additional simulations are not needed unless the physical interaction with hardware (mouse, keyboard, etc.) is also required – which is usually not the case. The logical control representations typically used are statecharts and, to a lesser extent, Petri nets, and STDS [248-251]. The dominance of statecharts in this application area can possibly be attributed to the fact that they have become a standard representation in the unified modelling language UML [252].

Beyond the application to software engineering, the majority of the scenario-based control approaches have been developed for computer animations in training, gaming, and entertainment. In these approaches movements are simulated based on key framing, i.e., on predefined frame sequences rather than on simulated physics [253]. In the same application area, the 'Iowa driving simulator' [254] used a more advanced approach. It projected virtual humans driving around in virtual cars, and it performed physics simulation controlled by combined scenarios specified as statecharts. In contrast to the approach in software prototyping, these covered not only human decision-making, but also the control by artefact subsystems.

In not all the approaches where scenarios are used in controlling simulations, scenarios have the same content. In their strictest sense, scenarios are restricted to conjecture about user actions. This interpretation is seconded by some approaches [253,255]. Other approaches do not explicitly distinguish human control from artefact control in scenarios [e.g., 249], they include actions by other humans (e.g., other traffic participants in the Iowa driving simulator), and/or they generate instructions for humans in the loop [256].

So far, in the support of product design, the application of scenarios as a means to control simulations of use processes with inclusion of human motor control and physical interaction has been scarce. In the investigated literature only two references to the application of scenarios in this area could be found. The first was in a side remark on possible implementation in the Santos virtual human (see end of 2.5.2). In the other, Hou et al. [255] claimed to have implemented scenarios to simulations of human-product interaction. These were rep-

resented as linear sequences of production rules (not as organized sets), and little detail is given about (i) the content of scenarios, (ii) the further capabilities of the simulation system, and (iii) the stage of development. No reports on further developments after 2007 could be found.

2.5.6 Discussion regarding control of simulations and simulation of control

From the viewpoint of application to simulations of human-artefact interaction, control should be considered in the following respects. Firstly, many products include control mechanisms. These can either be continuous or discrete. A simulation approach that supports comprehensive simulation of use processes should include this control. For modelling and simulation of continuous control in artefacts, the industry typically uses block diagrams, which therefore can be considered the preferred representation form. For modelling and emulation of discrete control several representations are in use, which are to some extent interchangeable. Graphical representations are gaining popularity over text-based descriptions because they are easier to use and to comprehend. Outside the specific area of manufacturing logistics, where Petri nets prevail, statecharts seem to be becoming the dominant graphical representation form in the industry.

Secondly, humans apply control in their interactions with products. Since human control applies to interaction with virtually any product, including it in simulations is even more essential than the inclusion of control embedded in products. The control manifests itself chiefly in motor control, i.e., control of muscles in order to move limbs and other body parts. Human-motor scientists have distinguished high-level control, which relies on cognition, and low-level control, which is subconscious and relies on eye-limb coordination. In many human simulation approaches, high-level control (and in some cases also low-level control) is not simulated but performed by a human in the loop.

High-level control can also be simulated, based on cognitive architectures such as ACT-R. Alternatively it may be substituted by a conjecture of human decision-making that is specified as a scenario specification, and executed by software in the loop. So far, the use of scenario specifications to control use-process simulations has largely been restricted to the field of human-computer interaction, where motor control has not been considered. Low-level control in existing human simulation approaches is, like high-level control, performed by a human in the loop or simulated. An alternative is to use pre-recorded motion-capture frames or animation frames. Since recently, the problem of simulating low-level motion control appears to have been solved. Several researchers have claimed that their human models can simulate low-level control of arbitrary changes in human posture (between specified start and end postures) with a high level of fidelity [e.g., 191,243].

As was also observed in 2.3.1 for equation-based models of humans and artefacts, the various human models discussed in this subchapter are universal: through parameterization they can be adapted to represent different individuals. Compared to creating artefact models, which have to be created from scratch

Table 2. Review of integral human simulations based on their capabilities

Integral human simulation approaches	Simulation capabilities									
	Range of behaviours covered					control-related functionality				
	rigid-body kinetics		deformations		kinematics	non-mechanical	control in artefacts	closed loop between human control and human motion behaviour	closed loop between human control and artefact behaviour	inclusion of high-level human motor control
	H	A	H	A	H	A	H/A	A		
SAMMIE, HECAD, COMBIMAN, etc. [Feyen et al.]										human in the loop
LifeModeler, Anybody									*b	motion capture frames
[Nedel & Thalmann]				*c				*d		animation frames or manipulation by user
Santos [Abdel-Malek et al.] +ACT-R/DHM [Carruth et al.]									*e	cognitive simulation (ACT-R)
Endorphin (*a)								*d	*b	timeline with animation keyframes, 'events', and 'behaviours'
Jack / Vis Jack [Badler et al.]		*f								human in the loop
[Hou et al.]	?	?						?	?	linear scenario, text-based



supported

supported with limitations (see notes)

unsupported

not applicable

unknown

H: human

A: artefact

*a: intended for entertainment industry, not for product evaluation. *b: only insofar interaction with artefact does not conflict with prescribed motion. *c: visualization only.

*d: human motor control: low-level only. *e: artefacts included for orientation and vision only.

*f: kinetostatics only.

references: Feyen et al. [119] LifeModeler [118], Anybody [117], Nedel & Thalmann [161], Abdel-Malek et al. [193], Carruth et al. [241], Endorphin [199], Jack [239], Hou et al. [255]

for each new product, creating human models is more lucrative because they can be more effectively reused. This may explain the integration efforts that have been put into building human models in particular. In this subchapter, several integrated human models have passed in review that also incorporate behavioural simulation capabilities as they have been discussed in 2.3. Most of these models have intentionally been developed to offer use-process simulation as a form of design support.

To wrap up the review of control, I have subjected these models to a closer inspection in order to assess the state of the art in integrating control and various physics behaviours in human-artefact interaction simulation. This outline will serve as a starting point for the final conclusions of the knowledge exploration in 2.6.

The overview in Table 2 brings together the most advanced human simulation models that were reviewed in this subchapter as well as some integrated

models that were reviewed in 2.3 (first two rows). There are two main groups of capabilities of these models that turned out to be discriminating:

- *Range of behaviours covered.* All of the models support simulations of human-body kinematics and none of them support simulation of non-mechanical behaviour or embedded control in artefacts. Most support rigid-body kinetics of humans and kinematics of artefacts. Full rigid-body kinetics of artefacts and, in particular, deformation is poorly supported by the models.
- *Supporting closed control loops.* In several cases, models fail to offer comprehensive simulations because they rely on (partially) open loops²⁶. In some of the models, the simulation of human control does not use feedback from the human motions resulting from it, because the control is entirely predefined. Two models²⁷ include this feedback only for low-level control by simulating aspects of eye-limb control. In two models this feedback is included up till the level of decision-making, however by deploying a human in the loop rather than by simulating it. In only one model this loop is closed up till the level of decision-making through cognitive simulation. It means that decisions can change the course of the simulated use process. This can also be realized by scenarios – whether these have been implemented by Hou et al. remains uncertain.

Another control loop is the one between human control and artefact behaviour. If the human control simulation cannot react on computed behaviours of artefacts, the simulation cannot be comprehensive. This is only supported by two models that rely on predefined motion sequences, but only if the artefact behaviours do not conflict with the predefined motion. (For instance: a ball striking a moving arm will bounce, but if a car strikes a moving arm, the arm will move through the car because its motion has been predefined). At present, there is no software-in-the-loop simulation approach that closes the control loop by feeding the effects of artefact behaviours back to (simulation or execution of) human decision making.

2.6 Discussion of the findings of the literature study

2.6.1 State of the art in simulations related to use processes

A large number of computer-based approaches are available to model humans, products and surroundings and to simulate their operation and behaviours without requiring deployment of human subjects. Existing simulation approaches appear to offer solutions that cover different patches of the problem area, which is diverse in several respects. This makes it possible to investigate partial aspects

²⁶ The issues related to open control loops in simulation are analogous to the issues related to loose coupling between behavioural simulations as was discussed in 2.3.7.

²⁷ The standalone version of Santos would actually be the third model in this category.

of use but comprehensive simulations that include many aspects concurrently are not possible. The currently available software falls apart into two categories: (i) engineering simulation software which is typically used for simulations of artefacts, parts of the human body, and interaction between artefacts and parts of the human body, and (ii) integral human simulation models, which have been built up around a parameterized model of the human body, and some of which permit inclusion of artefact models in the simulation.

Following from the diversity and complexity of use processes, there are three issues in simulation that need attention because they have been insufficiently addressed by those simulation approaches that form the current state of the art. These issues can all be considered integration issues.

Firstly, use processes involve various types of behaviour that can be categorized based on the areas of physics (mechanics, thermodynamics, optics, etc.). Most of these physical behaviours can be observed in both humans and artefacts, while some of them are more commonly observed in humans or artefacts specifically. To simulate these behaviours, behavioural simulation approaches have been developed. None of the currently available behavioural simulation approaches seems to be able to simulate all these types of behaviours. In engineering, mechanical multibody dynamics simulation software and finite-elements (FE) simulation software are the most commonly used.

The most versatile approaches that are well-suited to application in design are the ones based on discretized 3D volumetric models (polygon mesh-based, which includes FE, and meshless). These cover all types of physical behaviours except kinematics²⁸. Most of the physics-oriented behavioural human simulation models focus on mechanical multibody dynamics, while some are limited to kinematics. It should be noted that it may not be necessary to include all the known physical phenomena in simulations of human-product interaction. Simulation of non-mechanical physical behaviour in use processes is not frequently addressed in the studied literature, and is perhaps not considered to be of great importance. Conversely, simulation of mechanical behaviour is a recurring topic in the literature.

Secondly, use processes involve various control mechanisms in humans and artefacts that influence behaviours by modifying constraints over time. Typically, these control mechanisms operate in closed loops, i.e., they change constraints based on detected effects of behaviours. Both in humans and in artefacts, continuous and discrete (or logical) control mechanisms must be distinguished. In artefacts, analogue control is continuous and digital control is logical. In humans, continuous control is typically associated with low-level motor adjustments and logical control with reasoning and decision-making by the brain²⁹. Both continu-

²⁸ Concurrent simulation of multiple behaviours is also essential. Below, this is discussed separately as the third issue.

²⁹ Whether the intermediate control mechanisms connecting the brain and the low-level adjustments are discrete or continuous is still under debate, but it can be assumed that at some point there is a transition between the two forms of control.

ous and discrete control can be included in simulations. Continuous control can be regarded as physical behaviour and it is usually simulated using behavioural simulation software. Block diagrams are commonly used. Logical control can be specified using procedural language or one of the various more user-friendly graphical representations for finite automata. These specifications can be executed in connection with behavioural simulations.

In particular, the inclusion of human control in simulations has been a topic of intensive research in recent years. Regarding the inclusion of continuous low-level human control in simulations, researchers have recently claimed achieving a high level of fidelity using optimization-based algorithms to predict posture changes. To include cognitive control, the two approaches that have been proposed are (i) simulation based on cognitive architectures and (ii) execution of instructions that have been specified in formal scenarios. Both have been applied to virtual prototyping in the specific domain of human-computer interaction, where motor control is typically bypassed in simulations. However, only with the first approach some success has been reported in application to other forms of human-product interaction. A disadvantage of cognitive architectures is that adding interactions to the production system requires laborious rule-programming in a procedural language.

The third issue is concurrent simulation/execution of multiple behaviours and multiple forms of control. In the real world, different behaviours continuously influence each other and behaviours and control influence each other mutually. Concurrent simulation of all the relevant behaviours and forms of control is essential to achieve 'complete-picture' simulations of use. Several integrated approaches have been proposed to achieve some level of concurrency (in behavioural simulation: 'multiphysics'), but no attempt has been made to connect all the relevant behaviours and forms of control. To date, for some relevant combinations the integration efforts have been based on simplifications, in particular by applying open simulation loops and open control loops.

Open simulation loops have been applied to reduce computation times in co-simulations of different behaviours with discretized and/or rigid-body 3D models. This is known as *loosely coupled* multiphysics simulation. A good example is co-simulation of deformations and kinetics, in which an FE-simulation computes stresses and deformations based on input from a multibody simulation, but deformations computed by the FE-simulation are not exploited to change geometries in the multibody model. *Close coupling* of simulations is avoided because it is too computation-intensive, but in this example it would be useful in the particular context of human-product interaction, where large deformations can often not be ignored.

Analogous simplifications have been applied to inclusion of control in simulations, especially in the investigated integrated human models (Table 2). In simulations with these models, the effects of simulated behaviour of limbs and artefacts were not fed back to (simulations of) human control, or to low-level motor

control only. Full inclusion of this feedback would imply perception simulation, which has actually been realized in one of the human models³⁰. However, work-arounds have also been proposed to close the loop without simulating perception – for instance by using so-called invariants of human motion.

2.6.2 Towards a new approach for simulating product use processes

The major issue in use-process simulation is to resolve the trade-off between minimizing the efforts of model-making and computation on the one hand, and maximizing the amount of information obtained from simulations for improvement of the product and its use interaction on the other hand. Minimizing the efforts of model-making can be achieved by reducing the need for creating specialized simulation models, i.e., by using simulation algorithms that can work with the product models used in design (currently CAD models) and with human models that do not have to be created by the designer, but which can be instantiated from a prepared library of human-body models. Minimizing the computation effort can be realized by using simple or simplified models, or by using advanced simulation algorithms. This thesis does not comprehensively address resolving this issue because it focuses on methodological issues.

Maximizing the amount of useful information obtained from simulations can be realized by integrating as many as possible aspects that are currently covered by separate modelling and simulation approaches. In the following exploration of opportunities to develop a new approach I have focused on the integration issues, while keeping in mind the modelling effort required for the designer.

Regarding the first issue mentioned in 2.6.1, supporting simulations of multiple types of behaviours in physics, it seems to be a good idea to prioritize the inclusion of mechanical behaviours, which appear to be the most dominant in typical human-product interactions. The inclusion of other phenomena should then be kept in mind as a next step. Integration of mechanical behaviours implies the inclusion of rigid-body mechanics (including kinematics) *and* large deformations. Since none of the existing human-product interaction simulations fully covers these two phenomena, the challenge at hand still seems considerable. Full coverage would also require some form of close coupling, which refers to the simulation aspects of the third issue mentioned in 2.6.1. The commonly used approaches to simulate these behaviours are based on different models: rigid-body mechanics is simulated with rigid 3D volumetric models, while deformations are simulated with discretized 3D volumetric models, in particular with polygon mesh-based models (FE).

Of course the solution could be to implement close coupling between FE simulations and multibody simulations, which implies that the ‘missing link’ in the existing loosely-coupled approach should be filled in. In other words: after

³⁰ In this approach (Santos + ACT-R/DHM), dynamic behaviour of artefacts was not included.

each simulation time step of the FE simulation, all the rigid bodies in the multi-body simulation must be re-instantiated based on the geometric changes computed by the FE simulation. This is a computationally intensive task, which requires replacement of volumetrically modelled parts by updated volumetric conversions of the meshed representations of those parts. Since the meshed representation of a part must actually be represented in its deformed state, remeshing is most probably needed at each simulation step. Close coupling also requires repeated recomputation of centres of mass, locations of joints, etc., and evaluation of possible inertia effects caused by changes in geometry. If these complications are taken into account, this solution does not seem to be attractive. It may explain why this approach has not been implemented in current simulation systems. Looking at Table 1, an alternative approach could be to use particle-based models. These models can be used to simulate rigid-body kinetics, kinematics, and large deformations concurrently.

However, with the exception of the discrete-flexibility approach, current implementations of particle-based simulations seem to have been developed for application in the entertainment industry, where achieving realistic simulation of physical phenomena is not needed. Another point of concern is that so far, particle-based models (including discrete flexibility) have not been developed towards inclusion of non-mechanical physical phenomena. As was postulated in 1.6, a possible exception is nucleus-based modelling [257]. This conceptual-design oriented modelling approach, which is currently at an early stage of development, allows multiphysics simulation using particle-based discretized geometries and, if needed, it can also incorporate non-discretized modelling elements.

Regarding the inclusion of control in human-product interaction simulations (second issue in 2.6.1), the largest unresolved problem appears to be to incorporate human decision-making as a form of control. Some progress has been reported on applying cognitive simulation methods to human-product interaction in a way that enables motor control. However, the scenario-based approaches that have been used in human-computer interaction seem to be more advantageous in application in design.

Scenarios can be implemented using graphical representations which are easier to use and to comprehend than the procedural language-based representations that are needed for cognitive simulations, and they offer more freedom to designers to 'play' with possible alternative unfoldings of use processes. Graphical logical representations also allow designers to 'bundle' scenarios into networks corresponding to multiple possible courses of use processes. Although informal scenarios have been widely applied in various design projects, the currently known formal scenario-based approaches have focused on human-computer interaction, and they do not address human motor control and physical interaction between humans and products.

Based on the findings related to high-level human control, the research questions (iii) and (iv) in 1.5 can be reformulated and unified as: *"Can an approach prescribing the use of such modelling principle offer control mechanisms to simulate use as one holistic process, based on designer-conjectured organized*

sets of use scenarios?”. If the scenarios serve as a substitute for models of human decision-making, they should be scenarios in the strictest sense, describing actions by the (virtual) human user of the product only.

Regarding the achievement of concurrency of behavioural simulations and control (third issue in 2.6.1), the aspect of closing simulation loops has already been addressed above, but solutions for closing control loops have not been discussed yet. What seems to be missing here is a scheme to close control loops in connection with simulations. The human control in simulations should receive (and use) feedback from its own effects that are detectable in movements of limbs and in the physical effects of interacting with artefacts. Existing human-artefact interaction simulation approaches do not address embedded control in artefacts. However, insofar virtual artefacts have been equipped with embedded control mechanisms, these should be coupled to the simulation in a way that is similar to the inclusion of human control. Actually, these two closed loops have been included in the reasoning model that was proposed in Figure 4 (p. 9). This reasoning model appears to be a good starting point for realizing adequately closed control loops in human-artefact interaction simulations.

3 CONCEPTS AND THEORY OF RESOURCE-INTEGRATED INTERACTION SIMULATION

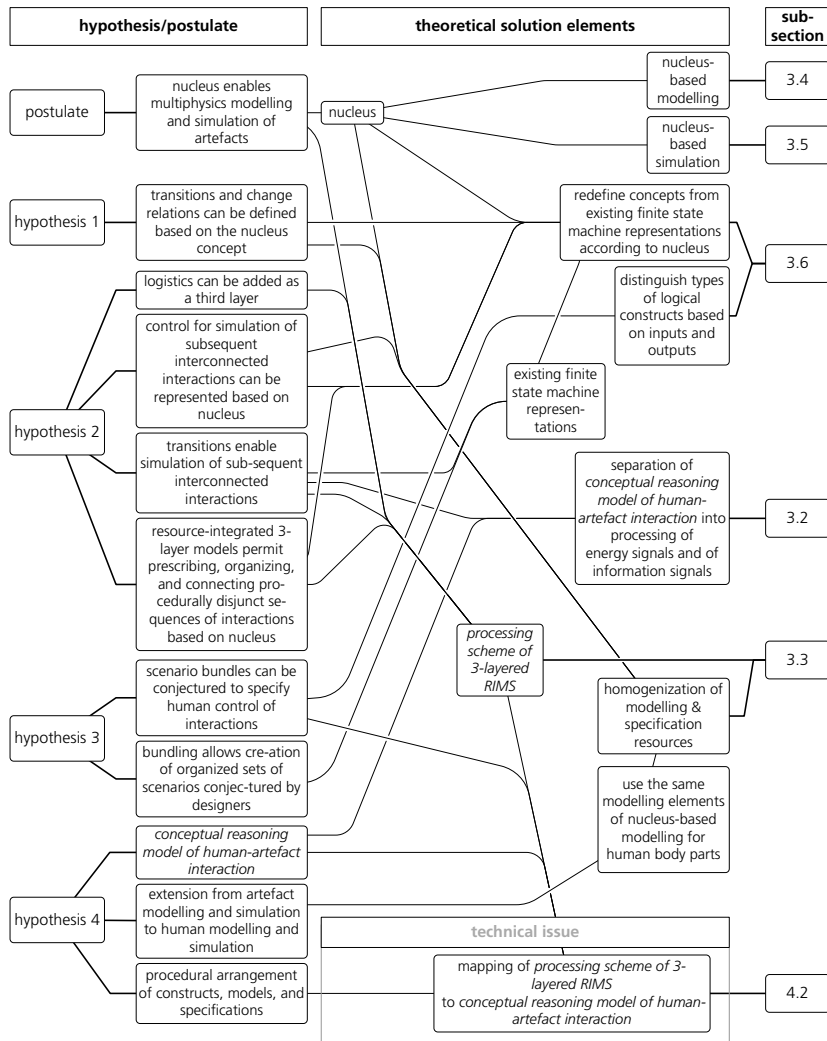
3.1 Introduction

In brief, my theory proposes a new approach for interaction simulation, applying the *nucleus concept* not only in entities modelling and behavioural simulation, but also in controlling the process of simulation. This forms a logistic layer above the basic layers of entity modelling and behaviour. The primary purpose of the logistics layer is to enable specification of scenario bundles. These allow designers to specify multiple sets of series of conjectured transitions caused by a fictitious end user of the product, who is making decisions that control his interactions based on perceived events during use.

The theory directly substantiates my respective hypotheses through the following elements: (i) adding transition relations to nucleus-based modelling and simulation, (ii) a three-layered concept for resource-integrated modelling and simulation, (iii) simulation control based on scenario bundles, and (iv) connecting control mechanisms and simulations according to a hypothesized conceptual reasoning model about human-artefact interaction. Working with the constructs that are based on these elements, the designer can do what-if type simulations and investigate the effects of varying the content of scenario bundles, models of the human user, and product models.

In this chapter, the constituents of the theory are explained starting out from the conceptual reasoning model of human-artefact interaction (section 3.2). Subsequently, the theory is presented regarding the principles of three-layered resource-integrated modelling and simulation (3.3), the postulated principles of nucleus-based modelling and simulation (3.4), the principles of nucleus-based behavioural simulation, (3.5), and the principles of simulation control by the logistics layer (3.6).

Figure 20 shows how the hypotheses have been decomposed and how the majority has been connected to the theoretical solution elements elaborated in the respective subchapters. An exception is the procedural arrangement of models and specifications. This is a technical rather than a theoretical issue ('how to' and not 'what to'), and thus it has been treated as an implementation issue in the next chapter.



Associative connections are to be read from left to right.
 'Processing scheme of 3-layered RIMS' refers to Figure 21 on page 58.
 'Conceptual reasoning model of human-artefact interaction' refers to Figure 4 on page 9.

Figure 20. Mapping of hypotheses to theoretical solution elements.

3.2 Conversion of the conceptual reasoning model of human-artefact interaction to a computational concept for simulation of interactions

Figure 4 on page 9 shows the proposed conceptual reasoning model of human-artefact interaction. In this model, the functionalities of humans and artefacts have been grouped by separating the processing of energy flows and the

processing of information flows. The reasoning model is computationally operationalized by (i) processing energy flows through simulation, based on mathematical models reflecting the actual physical phenomena, and (ii) processing information flows by executing logical instructions that reflect a syntactical interpretation of the physical phenomena underlying the signals. The execution of logic-based information processing acts as a control mechanism over simulation of energy-flow processing, which governs the physical interaction between humans and artefacts. To enable the control mechanism, it is interfaced with the simulation. The interfacing requires a two-way conversion: (i) of signals representing energy to an interpretation as information, and (ii) of logical instructions to physical variables that effectuate energy flows. The first conversion corresponds to sensing (perception in humans and sensor input to artefacts) and the second one to activation of effectors (muscles and actuators, respectively).

In this thesis, simulation of physical processes is limited to mechanics. Nevertheless, this makes it possible to investigate a wide range of interactions in which human body parts are engaged with artefacts through motion and force exertion. In addition to these mechanical interactions, the reasoning model allows consideration of information exchange between artefacts and humans through non-mechanical processes. This is achieved through a direct connection between specifications of human and artefactual logic, which disregards the conversion of information to energy (e.g., light or sound) and back to information, as it is common practice in established approaches to human-computer interaction based on cognitive architectures such as ACT-R and EPIC [229,230]. Examples of such forms of information exchange are displaying information visually on a monitor (information from artefact to human) or data entry through speech recognition (information from human to artefact).

3.3 Resource-integrated modelling and simulation

Modelling and specification of human-artefact systems needs various sets of entities that are defined and instantiated by the user and processed by the computer. These entities are actually available means that allow us to represent and process chunks of knowledge in a specific context. As such, they can be considered and called resources. In order to reduce the variety of elements needed, my aim has been to homogenize the resources involved.

As a unifying modelling approach, the nucleus concept has been chosen because of its ability to enable multiphysics simulation of artefacts and to allow designers to perform simulations with incomplete models during conceptual design. To extend homogenization of resources to the third layer, the nucleus concept had to be broadened (i) to represent not only artefacts in the metric space but also human users, and (ii) to process not only behavioural relations between entities in the metric space (in order to compute continuous simulations) but also logical relations between entities that are arranged in the time space to control the simulations.

We assert that nucleus-based models can be used to represent the human

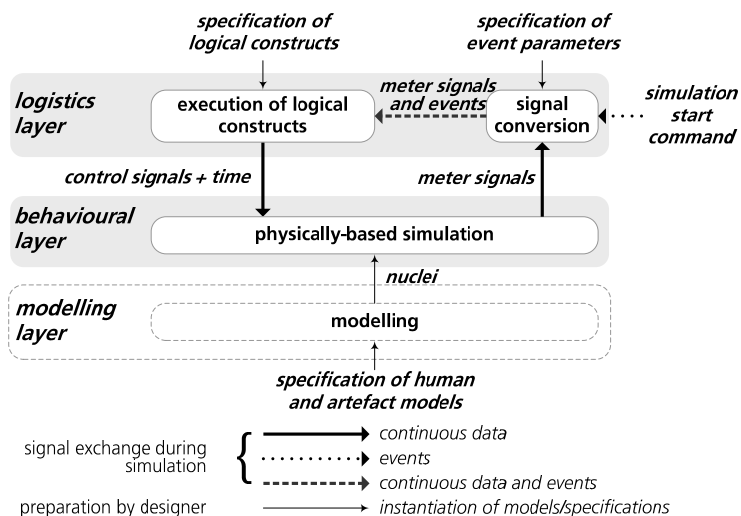


Figure 21. Processing scheme of three-layered resource-integrated modelling and simulation

body in the metric space without modifications to its underlying theory³¹, which is summarized in 3.4.

The second aspect of resource integration is representation and handling of logic. This has not been embedded in the theory of nucleus-based modelling so far. Its consequences in terms of formal definitions have been elaborated in 3.6.

Nucleus-based modelling and simulation has originally been proposed as a two-layered approach, involving a modelling layer and a simulation layer on top of it. As was hypothesized in Figure 3 (p. 8), representation and handling of logic necessitates a third layer on top of the simulation layer. Figure 21 shows how the third layer is included in terms of computational processing. The third layer does not have direct connections to the modelling layer. Since the result of modelling is available before running simulations and since development of the modelling layer is not the focus of this thesis, it has been drawn in dashed lines. As shown in Figure 21, logistics as a third layer calls for the specification of *meter signals*, *control signals*, and *events*, to connect it to behavioural simulation as will be explained in 3.6

In the next sections, I will elaborate on the underpinning theory for models, specifications, and processing algorithms of the logistics layer, which is the focus of this thesis. Before that, the underpinning theory of the simulation layer, with which it interfaces intensively, and, first of all, the modelling layer, which provides the relations of which the behaviour is to be simulated, are elaborated insofar as needed. The nucleus concept provides a suitable theory to underpin resource-integrated modelling and simulation, but it needed further research to learn how

³¹ This assertion follows from the literature survey in Chapter 2, where it was shown that in several cases common artefact modelling techniques have been used for human body modelling without modification of the underlying theories.

to extend it toward control without hurting the internal consistency of its theory. A kind of holism can be achieved when this concept is projected to the main constituents

3.4 Nucleus-based representation of humans and artefacts in the modelling layer

We³² applied *nucleus-based modelling* [28] with the assumption that it helps to overcome the disadvantages of the conventional volumetric and discretized physical models as described in Chapter 2. Nucleus-based modelling allows, among other things, describing large deformations of flexible objects, if the constitutional equations for the given material and loading conditions are specified in the relations.

3.4.1 Relation specification and management issues

The nucleus theory is seen as a foundational theory of a computer-oriented product modelling methodology. Obviously, this novel approach to conceptual and detailed modelling of products introduces new concepts, notions, terms and words that need to be defined, explained and put into context. The nucleus has been hypothesized as a new modelling entity focusing on *design concepts* that are intuitively or systematically generated by the designers and on making it possible to represent their elements and entirety. Below we explain the fundamental ideas and clarify the specific notions. We had investigated various engineering products and found that they all can ultimately be decomposed to a purposeful composition of physically coupled pairs. Any physically coupled pair can be abstracted as a composition of – typically two – interacting objects and multiple physical relations between the objects that may appear in various situations. Actually, this abstract construct gave the idea of the nucleus, which is understood as a generic modelling pattern that can be specialized to describe the constituents of a design concept or its entirety.

Representation of a most elementary design concept requires at least one nucleus. Compound design concepts however need a purposeful composition of a finite number of nuclei. In a sense, a nucleus is an abstraction of a physically coupled pair towards a formally (mathematically and computationally) underpinned way of modelling. From a programming point of view, a nucleus is a complex data and relation structure that covers geometric, structural, morphological, material and physical aspects. From a modelling point of view, this is the lowest level entity that carries both morphological and functional information to applications through the embedded structure of objects, relations, and conditions.

As mentioned above, our intention has been to represent design concepts

³² The theory in 3.4 refers to ongoing research in the CADE section. To express that it is based on collective efforts, it is discussed in the ‘we’ form.

by a purposeful set and configuration of nuclei. With symbolic terms, we formalized a design concept as $DC=\{O,\phi,S,C,A,D,P\}$, where O a set of pairs of objects, A =attributes of objects, ϕ =physical relations, P =parameters describing the relations, S =situation in space and time, D =descriptors of situation, C =constraints on attributes, parameters, and descriptors. Both nuclei and design concepts can be considered at five different levels, ranging from particle to system (see 3.4.3). Depending on the abstraction level, they can be decomposed but not beyond any limit. The objects, relations, and situations are minimal conceptual constructs in line with the human cognition units playing a role in intuition and memory recall. If they are missing, the abstraction becomes meaningless. Actually, this is another reason to call the triplet $N=\{O,\phi,S\}$ the nucleus of a design concept (Figure 22, [28]). A semantics-driven decomposition of a design concept results in nuclei at particle level that represent its ultimate constituents.

The objects in a set of relations are arranged in a *situation*, which creates a given structure of elementary changes described by the mathematical formulas. As will be explained in 3.5, a situated nucleus lends itself to a computable construct. From the aspect of simulation it means that temporal changes in the parameter values are represented by and computed based on the mathematical formulae and constraints.

3.4.2 Geometric representation of objects as particles and constructs of particles

Nucleus-based modelling assigns relations either to one metric object or between two associated metric objects. Below we shortly explain the fundamental notions of the geometric modelling approach used in association with the nucleus concept. The geometry of an object is represented by discrete entities, called particles, π , that are positioned by their (position) reference vector, \mathbf{r} . The reference vector of π is a localized vector in \mathbb{R}^3 showing to π from the origin of the global reference frame I . Two sorts of particles are distinguished: boundary particles that represent the bounding surfaces of an object, and internal particles used to represent the volumetric properties of a geometric object. Both the boundary and the internal particles have the means to describe material properties assigned to them. An infinitesimal half space (HS) is also assigned to each boundary particle to show its surface normal vector and (principal) curvature properties around the reference point (i.e., around the end point of \mathbf{r}).

The reference point is always contained by the infinitesimal surface patch. Actually, these infinitely small

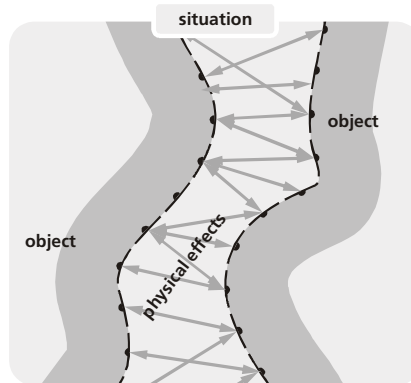


Figure 22. Ontological conceptualization of a nucleus

half spaces are used in building geometric models with *particle clouds*, Π , and *particle systems*, Π_s . Particle clouds can represent topologically open shells as discrete entity sets, whereas particle systems can represent connected internal domains as well as topologically closed shells as discrete entity sets. Both may have coverings, resulting in the *bounding surface* of a particle cloud or particle system. The covering of a particle cloud can be computed based on the geometric information conveyed by the half spaces of the adjacent boundary particles. A covering of part or whole of a particle cloud results in a bounding surface. More precisely, the covering corresponds to a proxy of the bounding natural surface of a modelled mechanical part of a product, and lends itself to effect carrying surface patches.

From a set of particles, a particle cloud Π can be composed that represents a part of a component. Since, from a geometric point of view, particle clouds are always incomplete, particle systems Π_s have been introduced to build complete components as structured compositions of sets of particle clouds. A particle system is complete if its boundary is closed. The three levels of geometric entities π , Π , and Π_s , are used to model geometry of products. If the considered system is extended to include elements other than the product, in particular humans and objects in the surroundings of the product, more levels need to be introduced to enable modelling and specification of technical systems and human-artefact systems.

3.4.3 Levels of modelling entities

Nucleus-based modelling extends to five levels of entity modelling, namely, particle, particle cloud, particle system, component, assembly, and system level. A nucleus can be instantiated on any level by defining physical couplings between reference points of entities on various levels. The entities of different levels have specific geometric meaning, as it is shown in Table 3 [24].

A component is a structured set of surfaces generated based on a particle system. An assembly contains a set of components of varying shapes generated based on a particle system and the kinematical relations in between. A system is an assembly-level composition of nuclei which represents artefacts and humans on various abstraction levels. During conceptualization, the various levels of modelling can be used to instantiate and represent objects on different levels. Detailing of parts is not a necessity in nucleus-based entity modelling. This gives the designer freedom in structure modelling.

Table 3. Geometric meanings of the levels of modelling entities

levels	<i>physical meaning</i>
particle	<i>infinitesimal surface and/or volume of the geometric construct</i>
particle cloud	<i>topologically open, finite surface and/or volume entities</i>
component	<i>topologically closed finite shape</i>
assembly	<i>shape aggregate with structural relationships</i>
system	<i>shape aggregation with functional relationships</i>

3.4.4 Relations

Relations of a nucleus are properties that express logical, geometric, and physical dependencies of two coupled objects. Relations are typically classified based on arity. Only unary ($1 \rightarrow$) and binary ($1 \leftrightarrow 1$) relations are considered based on the assumption that any $M \leftrightarrow N$ relation can be decomposed to M number of $1 \leftrightarrow N$ relations, which can further be decomposed to N number of $1 \leftrightarrow 1$ relations. Unary relations are reflexive and concern one object only.

Binary relations are defined between two objects. In the conceptual framework of nucleus-based modelling, three groups of reflexive relations (i.e. existence, reference and substance), and eight groups of binary relations (i.e. connectivity, positioning, morphological, kinematical, deformation, kinetic, physical phenomenon, and physical field) have been defined. These groups are subdivided into specific relationships, for instance 'mass' and 'surface normal vector' as subclasses of the substance relation, and 'friction' as a subclass of physical phenomenon relations.





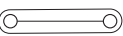


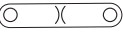
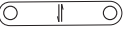
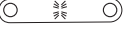


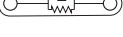

Note that some of the relations can be defined on each modelling level (e.g., connectivity), while others are specific to a given level (e.g., the surface normal vector relation, which can only be defined at particle level). The groups of the relations are also dependent on each other, since they use the information captured in their parent relation group. For instance, a morphological relation of two objects provides information about the degrees of freedom for kinematical relations. These examples show that nucleus-based modelling is a strongly relation-oriented modelling approach, in which relations play a more important role, and take priority over the metric entities that they logically, geometrically, and physically connect.

Another classification of relations can be introduced by taking into consideration the types of entities that are coupled in a nucleus. If two entities interconnected in a nucleus belong to the same component, then the nucleus is called internal, otherwise it is external. From this, three classes of relations can be derived: (a) internal relations (e.g., deformation), that can be specified only within particles of one particle system, (b) external relations (e.g., kinematical relations), that can be specified between particles of two particle systems and (c) hybrid relations (e.g., multiphysics), that may concern one particle system or a pair of particle systems.

With a nucleus it is possible to specify both unary relations and binary relations. A unary relation couples an object to itself, in order to describe reflexive properties such as mass. A binary relation couples two objects, which can be at the same level (e.g. two parts of the human body) or different levels (e.g., between a particle included in the model of the human body and in an assembly).

In this thesis, both the further development of the nucleus theory as well as its application to a proof-of-concept implementation have been targeted to investigation of mechanical behaviour of solid objects in use processes. Table 4 shows the key relations that I had to implement in order to allow for simulation of solid-mechanics behaviour.

Table 4. Key relations in solid mechanics

modelling element		2D symbol
entity (e.g., particle)		
relations	existence, reference (basic relations for every nucleus, defining its place in the data structure of the model)	(none)
	unary	
	substance	
	mass	
	surface	
	normal vector	
	morphological	
	differential description of half-space	
	connectivity (basic relation between two entities expressing its internal or external nature with respect to a higher-level entity)	(none)
	positioning (basic relation between every entity and a relative or absolute reference point defining its positioning in the 3D space)	(none)
	distance	
	kinematical	
	prescribed velocity	
	prescribed angular velocity	
	binary	
	morphological	
	facing	
	physical phenomenon	
	friction	
	collision	
	kinetic	
	prescribed force (actuator, muscle)	
	prescribed torque (actuator)	
	general	
	deformation	
	spring-damper (specifically used for stiff connections in human bones)	

At this moment in the discussion of the underpinning theories, it is necessary and possible to create a link to resource-integrated simulation of solid-mechanics processes. According to my reasoning, the connection to the logistics layer can be established by considering the prescribed velocity relation, the prescribed angular velocity relation, the prescribed force relation and the prescribed torque relation. These relations are described by mathematical functions containing parameters $p_i \in P$, with P the set of control variables. The control signals from the logistics layer contain the values for these parameters during simulations.

The prescribed-velocity relation of a referring object o_j relative to a reference object o_i is defined as:

$$\phi^{vp}_{ij} = \{o_i, o_j, \|vp_{ij}\|, x^{vp}_{ij}, y^{vp}_{ij}, z^{vp}_{ij}\}$$

where $\|vp_{ij}\| \in P$ is the magnitude of the prescribed velocity, and $x^{vp}_{ij}, y^{vp}_{ij}, z^{vp}_{ij} \in P$ are its non-normalized vector components. This results in a prescribed velocity of o_j relative to o_i equal to

$$\mathbf{vp}_{ij} = \begin{bmatrix} \frac{x_{ij}^{vp}}{\sqrt{(x_{ij}^{vp})^2 + (y_{ij}^{vp})^2 + (z_{ij}^{vp})^2}} \\ \frac{y_{ij}^{vp}}{\sqrt{(x_{ij}^{vp})^2 + (y_{ij}^{vp})^2 + (z_{ij}^{vp})^2}} \\ \frac{z_{ij}^{vp}}{\sqrt{(x_{ij}^{vp})^2 + (y_{ij}^{vp})^2 + (z_{ij}^{vp})^2}} \end{bmatrix}$$

If x_{ij}^{vp} , y_{ij}^{vp} , and z_{ij}^{vp} are not specified, then, by default,

$$\begin{bmatrix} x_{ij}^{vp} \\ y_{ij}^{vp} \\ z_{ij}^{vp} \end{bmatrix} = \mathbf{r}_j - \mathbf{r}_i,$$

where \mathbf{r}_i and \mathbf{r}_j are the reference vectors of o_i and o_j . This allows the designer (or other person who specifies parameters in the logistics layer) to specify a prescribed velocity along the line through the reference points of o_i and o_j by only its magnitude. Velocities in other directions can be specified independently of the magnitude by including the components.

Likewise, the prescribed-angular velocity relation, the prescribed-force relation and the prescribed-torque relation are defined as:

$$\phi^{\omega p}_{ij} = \{o_i, o_j, ||\omega p_{ij}||, x^{\omega p}_{ij}, y^{\omega p}_{ij}, z^{\omega p}_{ij}\},$$

$$\phi^{Fp}_{ij} = \{o_i, o_j, ||Fp_{ij}||, x^{Fp}_{ij}, y^{Fp}_{ij}, z^{Fp}_{ij}\},$$

and

$$\phi^{Mp}_{ij} = \{o_i, o_j, ||Mp_{ij}||, x^{Mp}_{ij}, y^{Mp}_{ij}, z^{Mp}_{ij}\}$$

respectively.

3.5 The behaviour layer: simulating behaviour with nucleus-based models

As described in 3.4, the physical relations imply elementary operation processes that manifest the behaviour of a nucleus. An aggregation of nuclei can also manifest the operation and behaviour of design concepts of arbitrary complexity. Actually, the time-dependent changes described by the physical relations will lend themselves to the abovementioned operations, or behaviour B of a nucleus, in predefined situations: $B(N) = G\{S_k(o_i, \phi_{ij}, o_j)\}$, where $o_i, o_j \in O$, ϕ_{ij} and S_k are as above, and G is a behaviour generator function, which takes into consideration the interaction of various nuclei and the influences on each other's behaviour. The introduction of G is necessary, since the observable operation of a modelled

design concept DC is an aggregation of the elementary operations of the nuclei. For the reason that all nuclei might interact in an assembly, this aggregation should be represented as a Cartesian product rather than as a Boolean union of the observable elementary operations, that is, $B(DC) = B(N_i) \times B(N_j)$, or $B(DC) = \Pi (B(N_i), B(N_j))$, where Π denotes a mathematical product.

The arrangement of situations, or in other words, the operation and interaction of the nuclei, are governed by so called scenarios. A scenario Σ prescribes a sequence of situations, in which the observable operation delivered by a nucleus or a configuration of nuclei incorporated in a design concept happens. That is, $\Sigma = \cup^*(S_k)$, where \cup^* denotes a sequentially arranged union. With these, the behaviour of a DC is: $B(DC) = G(\Sigma\{N_i\})$, or, on the level of relations, $B(DC) = G(\cup^*(S_k(o_i, \phi_{ij}, o_j)))$. Specification of the physical relations includes definition of the parameters, the mathematical formulas (equations and rules) that relate the parameters to each other, and the constraints and value domains. Thus, a nucleus is a primitive system in itself, since its data structure contains all pieces of information that are needed to simulate its behaviour.

A nucleus can be instantiated in different situations, which in turn means instantiation of the elementary operation processes in different forms. Not only complex design concepts, but also humans and objects in the surroundings of the product can be defined in the same way and deployed to simulate the interaction in a use process. Solid mechanics offers the means to treat the four main observable phenomena: motion, collision, deformation and fracture³³.

It is worth mentioning that phenomena relating thermodynamics, fluid dynamics, gas dynamics, and so forth can also be considered in relations. It is a fact however that there exists no single predictive model that is capable to incorporate all phenomena and interrelated changes, not even theoretically.

3.6 The logistics layer

As was explained in 3.4, in a physically-based simulation model a nucleus N is always considered in a situation during which the relations do not change. However, the situations change during a physical behavioural process. In simulations, the change of situations needs to be controlled but also the physical changes need to be computed. The changes of situations can be enabled and specified by introducing control over them. Towards this end, the following needs to be considered.

Mathematically, as long as the parameters $p_i \in \mathbb{R}$ in the relations ϕ_i do not change, a nucleus N describes one single situation S , but if, at a given point in time, the control changes one or more of these parameters p_i , the result is that S ends, and a different instantiation of N describes the content for a next situation S' . This could, for example, happen by human intervention during interaction or,

³³ For this thesis, fracture has not been considered in modelling solid mechanics behaviour.

in general, through a *transition* τ in the use process. A set of concurrent changes in terms of the parameters of the relations associated with a nucleus N is described by (or specified/scheduled in) a *change relation* $\delta = \delta(\tau, \{p_i(\phi(N))\}) = \{k_i\}$, where $k_i \in \mathbb{R}$ are the new values for p_i if the transition τ takes place.

A change relation is a relation between two situations that are subsequent in time. However, change relations do not explicitly address situations as identifiable entities, because a situation is based on the assumption that none of the parameters p_i in any of the ϕ_i in N changes during a particular interval in time. The validity of this assumption depends on computation results during a simulation (or on happenings in a real use process), therefore it is not known beforehand if or when a particular set of values $\{k_i\}$ will be applied.

In terms of the procedural implementation it can be concluded that the logistics layer L sends signals conveying changes in parameters to the simulation layer Ψ . They actually take the form of *control values* $p_i(t)$.

$$L : \{m_i(t)\} \rightarrow \{p_i(t)\},$$

where $m_i(t) \in \mathbb{R}$ are *meter values* produced by the algorithms of the simulation layer Ψ , which process control signals in order to perform the simulation and to generate meter values:

$$\Psi : \{p_i(t)\} \rightarrow \{m_i(t)\}.$$

The control signals form a subset of the parameters of instantiated relations in the simulation model, representing those parameters that specify control of muscles and actuators. Meter signals describe selected/specified physical quantities that can be ‘measured’ in the simulation output. Through evaluation of the meter signals, the logistics layer changes values of control signals to effectuate transitions τ_i that cause changes in situations. The time signal $t \in \{m_i(t)\}$, which is generated in the logistics layer, is a special meter signal because it is also transferred to the simulation together with the control signals. In the formal definitions presented in this subchapter, signals are considered as time dependent and this fact is expressed by using the argument ‘(t)’ for time-dependent variables. Note that all the meter variables other than t are regarded as mechanical variables. For this reason, their signals are called *mechanical meter signals*.

The logistics layer L has been formalized as $L = \{A, \Xi\}$ where A is the set of *logical constructs* and Ξ the *signal conversion specification*. The set of logical constructs $A = \{\lambda_j\}$ in the logistics layer L receives inputs in the form of meter events $e_{met,i}(t)$ and condition values $v_i(t)$, $\{v_i(t)\} \supseteq \{m_i(t)\}$, and it produces outputs in the form of control values $p_i(t)$ as well as *logistic events*³⁴ $e_{lgs,i}(t)$:

$$A : \{e_{met,i}(t)\}, \{v_i(t)\} \rightarrow \{p_i(t)\}, \{e_{lgs,i}(t)\}$$

The signal conversion specification Ξ receives inputs from the physically-based simulation Ψ in the form of all meter signals $\{m_i(t)\}$ and from the logical constructs A in the form of logistic values $\{u_i(t)\}$ and logistic events $e_{lgs,i}(t)$, and

³⁴ Events have been further defined in 3.6.2

produces outputs in the form of condition values $v_i(t)$ and meter events $e_{met,i}(t)$:

$$\Xi: \{m_i(t)\}, \{u_i(t)\}, \{e_{lgs,i}(t)\} \rightarrow \{v_i(t)\}, \{e_{met,i}(t)\}$$

In the next two sections I will elaborate on the formal definitions of the concepts populating the logistics layer. The logical constructs will be discussed in 3.6.1-3.6.4 and the signal conversion specification in 3.6.5.

3.6.1 Formal definition of logical constructs

Logical constructs contain the transitions that are the instantiation of nuclei in the time domain, and that serve as control elements. They have been defined to provide means for controlling physically-based simulations by changing parameters in relations. The formalization below involves various concepts known from finite automata representations which have also been applied in control of physics simulations and continuous physical processes based on logic. The involved key concepts are transitions, events, and states. The set of logical constructs \mathcal{A} consists of three³⁵ logical constructs $\mathcal{A} = \{\lambda_s, \lambda_\ell, \lambda_p\}$ where λ_s is the *scenario bundle*, λ_ℓ is the *model of low-level logical control of human motion* and λ_p is the *procedure structure* in which embedded software of artefacts is specified³⁶. These three logical constructs differ in the inputs they receive and in the outputs they produce.

The scenario bundle receives meter events, logistic values and events (from the procedure structure), and condition values, and it produces logistic events. This can formally be specified as follows:

$$\lambda_s: \{e_{met,i}(t)\}, \{u_i(\lambda_p, t)\}, \{e_{lgs,i}(\lambda_p, t)\}, \{v_j(t)\} \rightarrow \{e_{lgs,i}(\lambda_s, t)\}.$$

The model of low level logical human motor control receives logistic values and events from the scenario bundle together with condition values, and it produces meter signals:

$$\lambda_\ell: \{e_{lgs,i}(\lambda_s, t)\}, \{v_j(t)\} \rightarrow \{p_i(t)\}.$$

The procedure structure receives meter events, logistic values, and events from the scenario bundle, and condition values, and it produces meter signals:

$$\lambda_p: \{e_{met,i}(t)\}, \{u_i(\lambda_s, t)\}, \{e_{lgs,i}(\lambda_s, t)\}, \{v_j(t)\} \rightarrow \{p_i(t)\}.$$

Thus, the processing performed by λ_s and λ_ℓ together is analogous to the processing performed by λ_p alone. Actually, λ_s and λ_ℓ can be together considered as a human interaction construct λ_h in which all logical processing by the human has been unified:

³⁵ Note that it is possible to change these definitions to include more than three logical constructs. Actually, this is required if logical control by multiple humans and/or multiple distinct artefacts is to be simulated. However, this possibility has not been explored in this thesis.

³⁶ Many products have control mechanisms (e.g., embedded software) that activate actuators as a reaction to input from sensors. As was pointed out in 2.5.6, this aspect has been ignored by existing approaches for simulation of human-product interaction.

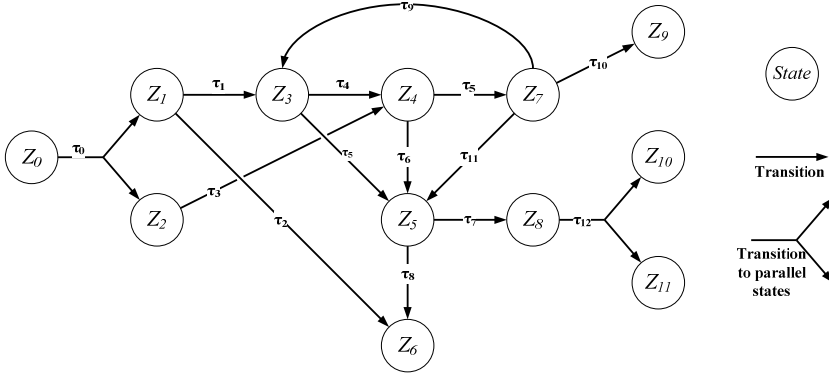


Figure 23. Example of a logical construct

$$\lambda_h: \{e_{met,i}(t)\}, \{u_i(\lambda_p, t)\}, \{e_{lgs,i}(\lambda_p, t)\}, \{v_j(t)\} \rightarrow \{p_i(t)\}.$$

A logical construct λ_j of the logistics layer is represented as a directed graph, in which the vertices are transitions τ_i and the nodes are states Z_i , i.e., $\lambda_j = \{\{\tau_i\}, \{Z_i\}\}$ (Figure 23). As will be explained in 3.6.4, transitions are placed at the beginning and ending of situations S_i which are specified by the change relations i that the transitions contain. As functions of time, τ_i and Z_i are Booleans, i.e., $\tau_i(t), Z_i(t) \in \{0,1\}$ where $\tau_i(t)=1$ means ' τ_i takes place', $\tau_i(t)=0$ means ' τ_i does not take place', and $Z_i(t) = 1$ means ' Z_i is active', $Z_i(t) = 0$ means ' Z_i is not active'.

Transitions between states are triggered by external events $e_{ext,i}(t)$, which the logical construct receives from the other constructs $\{\{\lambda_i\}, \Xi\} | i \neq j$. A special type of external event is the meter event $e_{met,i}(t)$, which recognizes the stimuli from the meter signals in time. In order to take place, a transition may require fulfilment of a logical condition γ_i expressing whether a particular event effectuates a transition or not, based on evaluation of specified meter values, logistic values and activeness of specified states. Together, the values evaluated in conditions are called *condition values*³⁷.

3.6.2 Definition of events

In general, an event $e_i(t)$ can be defined as the fact that a particular, measurable and predefined change in a process takes place. The change is predefined because it is considered of interest as a trigger to cause transitions in the logistics layer. Since it occurs at a point in time only, an event has no duration. Two distinctions between types of events are made: an event is either internal or external and it is either a meter event or a logistic event. An internal event reflects changes inside the logical construct under consideration, while an external event reflects changes in another logical construct.

³⁷ Note that in 3.6 it was stated that $\{v_i(t)\} \supseteq \{m_i(t)\}$, however only for those condition values $v_i(t)$ that are output of the signal conversion specification, it is specifically true that they are all meter values

Meter events $e_{met,i}(t)$ correspond to specified changes in meter variables, i.e., simulation outputs or time. They are specified in the signal conversion specification, and have been further defined in 3.6.5. All meter events are external.

Logistic events $e_{lgs,i}(\lambda_j, t)$ correspond to specified changes in a logical construct, i.e., typically the change that occurs because (i) a particular transition has taken place, or (ii) a particular state has become active or inactive. A logistic event is a consequence of a meter event occurring at the same time. These events have been introduced respectively to (i) synchronize transitions so that they take place at the same time, and (ii) to allow for a shorter description of the fact that an *arbitrary* transition of which the state in question is the target state (or source state, respectively) has taken place. A logistic event can be external or internal.

3.6.3 Definition of control variables

A control variable p_i is a means of manipulating a relation ϕ_i in the physically-based simulation model Ψ . It allows prescribed specific change of its value, which is used in computation of the behaviour. Typical examples of control variables are prescribed speeds in kinematical relations, or prescribed forces in kinetic relations, for instance to control actuators. Certain variables in relations cannot appear as control variables. For instance, masses in substance relations and Young's moduli in deformation relations cannot be considered as such.

Now let us consider the manipulations that the set of logical constructs \mathcal{A} performs on relations in Ψ as a result of processing meter events and condition values (which the signal conversion specification produces based on the meter values it receives from Ψ):

$$\mathcal{A}: \{e_{met,i}(t)\}, \{v_i(t)\} \rightarrow \{p_i(t)\},$$

thereby ignoring logistic events $\{e_{lgs,i}(t)\}$, since they are not output to Ψ . Now, if a meter event $e_{met,j}(t_j)$ occurs at $t = t_j$, this event *influences* a control variable $p_i(t)$ at $t = t_j$. This can formally be written as $I_j(p_i(t), t_j) = (1, k_j)$ where $I_j \in \{0, 1\} \times \mathbb{R}$ is the *influence descriptor* and $k_j \in \mathbb{R}$ is the *control variable modifier*, which is a constant specifying a new value for $p_i(t)$ (as defined below)³⁸. On the other hand, if the occurring event $e_{met,j}(t_j)$ does *not* influence $p_i(t)$ at $t = t_j$, then $I_j(p_i(t), t_j) = (0, 0)$. Whether and when a meter event influences a control variable depends on the transitions, conditions, states, and logistic events specified in the logical constructs as well as on the simulation results. Therefore, it is not known beforehand if particular meter events will influence particular control values.

The value of a control variable modifier k_j is specified as an instruction to be executed at a transition that takes place when $e_{met,j}(t_j)$ occurs³⁹. To describe how

³⁸ Here, the index i is used for vector components, j is used for indexing intervals in time, κ is used for relations, and J is used for nuclei.

³⁹ The transition does not have to be directly associated with $e_{met,j}(t_j)$. It can also be associated with a logistic event that follows from $e_{met,j}(t_j)$ and thus occurs at the same time t_j .

k_j modifies a control value $p_i(t)$, let us first define a control interval $K_i(c_i, t_j)$ between two successive transitions at $t=t_j$ and $t=t_{j+1}$ as

$$K_i(c_i, t_j) = [t_j, t_{j+1}) \text{ if } \begin{cases} l_j(p_i, t_j) = (1, k_j) \\ l_{j+1}(p_i, t_{j+1}) = (1, k_{j+1}) \\ \forall t | t_j < t \leq t_{j+1} : l(p_i, t) = (0, 0) \end{cases}$$

Now, $p_i(t)$ can be defined as a step function,

$$c_i(t) = \sum_{j=0}^{n_i} \chi_{K_j}(t)$$

where χ_{K_j} is the *indicator function*

$$\chi_{K_j}(t) = \begin{cases} k_j & \text{if } t \in K_j \\ 0 & \text{otherwise} \end{cases}$$

and n_i is the total number of times⁴⁰ that meter events influence a particular control value $p_i(t)$. Furthermore, for $j = 0$, $t = t_0$. At the start of the simulation, i.e., for all $t < t_0$, the meter events, meter values, and control values have initial values as follows:

- Meter events: $\forall i : e_{met,i}(t_0) = 0$
- Meter values: all initial values of meter variables $m_i(t)$ are initially specified in the physically-based simulation model. Typically these initial meter values correspond to a static state of the human-artefact system.
- Control values: the designer can specify initial control values $p_i(t)$ that are sent to the simulation layer \mathcal{P} before the start of the actual simulation.

3.6.4 Definition of transitions and change relations

A transition⁴¹ τ_i with condition γ_i connects a *source state*⁴² Z_i to one or more *target states* $Z_i(\tau_i)$, where $i \geq 1$. If $i > 1$, then $Z_i(\tau_i)$ are called parallel target states of τ_i . Whether a transition

$$\tau_i(t, e_i(t), \gamma_i(v_1(t), \dots, v_n(t)), \varepsilon_i(t))$$

takes place at a time t ($\tau_i(t) = 1$) depends on:

- a meter event or logistic event $e_i(t)$ which is said to *trigger* the transition if it actually takes place,
- a condition $i = \{g_i(v_1(t), \dots, v_n(t)), \diamond, c_i\}$, where $g_i: \mathbb{R}^n \rightarrow \mathbb{R}$ is a function of specified condition values $v_1(t), \dots, v_n(t)$, $\{ \cdot \} \supseteq \{m_i(t)\} \cup \{u_i(t)\} \cup \{Z_i(t)\}$, $\diamond \in \{=, \neq, <, >, \leq, \geq\}$ is a relational operator, and c a constant, $c_i \in \mathbb{R}$ so that $g_i \diamond c_i$, and

⁴⁰ Note that n_i depends on what has been specified in the logical constructs and also on the simulation results. Therefore, n_i is not known beforehand.

⁴¹ Here, the index i is used for transitions, i (iota) for parallel target states of a transition, and j is used for control variables.

⁴² In literature on finite automata, the terms *source state* and *target state* are commonly used for the two states connected by a transition.

- the enabling $\varepsilon(\tau_i, t)$ of the event: the transition is enabled, $\varepsilon(\tau_i, t) = 1$, when $Z_i(t) = 1$, i.e., that its source state is active.

so that

$$\tau_i(t) = \begin{cases} 1 & \text{if } e_i(t) \wedge \gamma_i(v_1(t), \dots, v_n(t)) \wedge \varepsilon(\tau_i, t) \\ 0 & \text{otherwise (i.e., the transition does not take place)} \end{cases}$$

with $\tau_i(t)$, $\gamma_i(t)$, $e_i(t)$, and $\varepsilon(\tau_i, t)$ Booleans, and $v_i(t) \in \mathbb{R}$. Since events have no duration, transitions do not have a duration either. As soon as a transition takes place, its source state Z_i becomes inactive and its target states $Z_i(\tau_i)$ become active.

A transition⁴³ τ assigns a set of modifier values k_j to control variables $p_j(\tau, t)$:

$$p_1(\tau, t) := k_1$$

⋮

$$p_j(\tau, t) := k_j$$

⋮

$$p_{m_p(\tau)}(\tau, t) := k_{m_p(\tau)}$$

where $1 \leq m_p(\tau) \leq n_p$, and $m_p(\tau)$ is the number of control variables modified by τ at t , and n_p the total number control values in the human-artefact system. In accordance with the definition of the step function for control values $p_j(\tau, t)$ above, the control values keep their newly modified values k_j until another transition takes place for which a value for k_j has been specified.

It has to be mentioned that although the set of control variables $\{p_j(\tau, t)\}$ that is manipulated by a particular transition τ corresponds to parameters p_j in the simulation layer Ψ , it is not necessarily true that these are all parameters of relations confined to one particular nucleus N . This means that a transition cannot generally be regarded as a relation that is strictly between two situations.

The example in Figure 24 illustrates how transitions interconnect the different sets of situations for different nuclei into a logical construct. In the figure, transitions have been specified that manipulate parameters in relations associated with two distinct nuclei N_1 and N_2 . The expression $\tau_i\{N_j\}$ stands for “*transition τ_i manipulates parameters in relations associated to a set of nuclei $\{N_j\}$* ” so that, for instance, $\tau_1\{N_1, N_2\}$ means that τ_1 manipulates control variables of relations associated with N_1 and control variables of relations associated with N_2 .

This particular example also shows the interconnection between states and situations, which is not straightforward either. Every situation except for the initial situation Z_0 contains one or more states, i.e., the example shows that a state does not generally correspond to a situation. Although transitions certainly provide the *connections between* situations, and *entering states* corresponds to *entering new situations*, the interconnections between transitions and states on the one hand, and situations in a simulation on the other hand are not as

⁴³ From here τ is written without index to avoid confusion with other indices. The following definitions apply to all transitions τ_i in a logical construct.

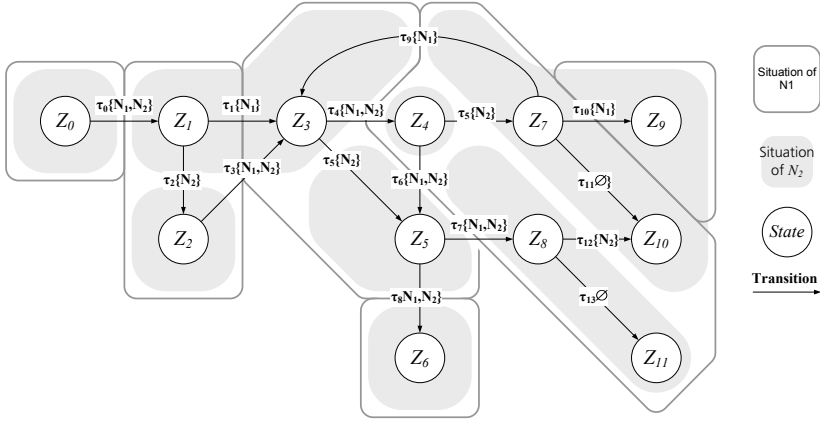


Figure 24. Example of transitions, situations, and states of a logical construct

straightforward as they may seem at a first glance.⁴⁴

To resolve the issue of complicated interconnections, *change relations* $\delta_i(\tau)$ are introduced to provide the actual interconnection with the notion of situations. To that end, the set of variables $\{p_j(\tau)\}$ of a transition is further subdivided into subsets $\{p_{\delta_i}(\tau)\}$ referring to change relations δ_i , with $i = 1..n_{\delta(\tau)}$, where $n_{\delta(\tau)}$ the number of change relations specified by τ ; $\cup\{p_{\delta_i}\} = \{p\}$; and $\cap\{p_{\delta_i}\} = \emptyset$, so that a transition holds multiple binary relations δ_i , each between two situations S_i and S_i' corresponding to subsequent instantiations of a particular nucleus N_i .

3.6.5 Formal definition of the signal conversion specification

The signal conversion specification (scs) is formally defined as $\Xi = \{\Xi_R, \zeta_0, \Xi_T\}$, where

- $\Xi_R = \{\zeta_{Rs}, \zeta_{R\ell}, \zeta_{Rp}\}$ are the specifications of stimulus recognition for human and artefact in the respective logical constructs λ_s , λ_ℓ , and λ_p ;
- ζ_0 the start event specification, and
- $\Xi_T = \{\zeta_{Ts}, \zeta_{T\ell}, \zeta_{Tp}\}$ are the specifications of human and artefact timing in the respective logical constructs λ_s , λ_ℓ , and λ_p .

The processing of signals performed by the signal conversion specification is written as

$$\Xi: \{m_i(t)\}, \{e_{igs,i}(t)\} \rightarrow \{v_j(t)\}, \{e_{met,i}(t)\},$$

in other words, the scs is responsible for:

⁴⁴ It should also be mentioned that situations cannot generally be mapped to the 2D graphical representation of a logical construct. Actually, Figure 24 shows a special case, where the assignment of states to situations does not depend on the path that is actually taken through the logical construct. However, if for instance the transition τ_6 would have been specified so that it only affects relations in N_2 , i.e., $\tau_6\{N_2\}$ instead of $\tau_6\{N_1, N_2\}$, the states Z_3 and Z_5 would belong to the same situation of N_1 if the direct transition τ_5 between Z_3 and Z_5 is taken, but not if the 'detour' $\tau_4 \rightarrow Z_4 \rightarrow \tau_3$ takes place. In that case the logical construct is still a valid one, but it no longer allows visualizing the boundaries of situations.

- (i) converting meter signals $m_i(t)$ from the simulation layer \mathcal{P} to meter events $e_{met,i}(t)$ that are input to the logical constructs \mathcal{A} . This is called ‘stimulus recognition’, referring to the human ability to recognize those stimuli from perceptive input that require action;
- (ii) generating the time signal t that is used by the logical constructs \mathcal{A} and the physically-based simulation, as well as the start event $e_{start} \in \{e_{met,i}\}$ and
- (iii) regulating the timing of logical constructs \mathcal{A} , i.e., hesitation and delays in the execution of control.

In the next two subsections, the processing related to (i), and to (ii) and (iii), respectively, has been formally elaborated.

Definition of stimulus recognition specifications and meter events.

The specifications of stimulus recognition Ξ_R convert meter signals, which are continuous functions of time, to meter events, which are discontinuous in time. $\xi_{Rj} = \{e_{met,i}(t), \{v_i(t)\}\}$ with $j = \{s, \ell, p\}$ and $\{v_i(t)\} \supseteq \{m_i(t)\}$ the set of condition values as has been defined in 3.6.4. The processing of meter variables to generate meter events is defined as follows:

Let $\{m_i(t)\}$ be the set of meter variables. Then we can define a subset of meter variables $\{\mu_i(t)\} \supseteq \{m_i(t)\}$ as *recognition variables*, which are used to specify meter events $e_{met,i}(t)$. Now we can define a meter event $e_{met,i}(t)$ as a Boolean

$$e_{met,i}(t) = e_{met,i}(r_i, h_i, f_i(t)) \in \{0, 1\}$$

where $r_i \in \{\uparrow, \downarrow, \updownarrow\}$ the orientation of the event: *increasing*, *decreasing* or *bidirectional*, $h_i \in \mathbb{R}$ is the *event threshold*, and $f_i(\mu_1(t), \mu_2(t), \dots, \mu_n(t)) : \mathbb{R}^n \rightarrow \mathbb{R}$ is the *event-generating function*. In the simplest case, the event-generating function reflects changes in the value of a single recognition variable, e.g., $f_i = \mu_i(t)$ (where μ_i could be for instance the distance between the human fingertip and a button that must be pushed), but it can also be an expression that requires evaluation of multiple recognition variables, e.g., $f_i = 5\mu_1(t) + \sqrt{\mu_2(t)} - \frac{d}{dt}(\mu_3(t))$. The meter event occurs if the value of the function $f_i(t)$ crosses the threshold in the direction(s) specified by the orientation (Figure 25).

This is formally defined as:

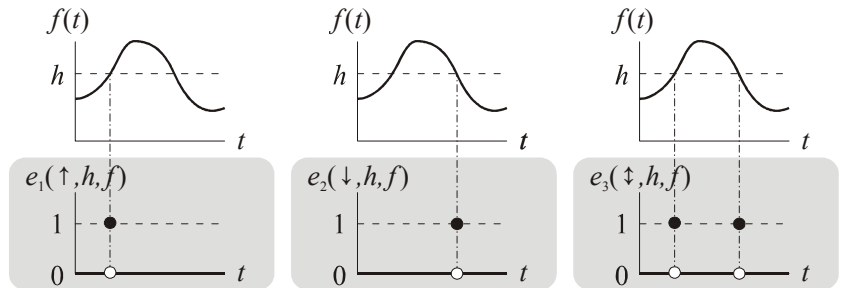


Figure 25. Occurrences of meter events e_1 with increasing orientation, e_2 with decreasing orientation, and e_3 with bidirectional orientation

$$e_{met,i}(t) = \begin{cases} 1 & \text{if } f_i(t) = h_i \wedge \left(\left(r_i = \uparrow \wedge \frac{df}{dt} > 0 \right) \vee \left(r_i = \downarrow \wedge \frac{df}{dt} < 0 \right) \vee \left(r_i = \updownarrow \wedge \frac{df}{dt} \neq 0 \right) \right) \\ 0 & \text{otherwise} \end{cases}$$

meaning that event $e_{met,i}(t)$ occurs *only* if its orientation is ‘increasing’ and the event-generating function crosses the threshold while increasing, or if its orientation is ‘decreasing’ and the event-generating function crosses the threshold while decreasing, or if its orientation is ‘bidirectional’ and the event-generating function crosses the threshold while changing in either direction.

Definition of the start event specification, the time signal and specifications of human and artefact timing. In the start event specification ζ_0 only one event is specified, namely, the start event $e_{start}(\uparrow, t_0, t)$, which is a special meter event that is instantiated to start the simulation at $t = t_0$. The start event specification prescribes that when the system user (designer) enters a start command, the start event e_{start} is sent to the scenario bundle and to the procedure structure. At the same time, the time signal is set to $t = t_0$. The scenario bundle and the procedure structure must contain an initial transition $\tau_0(e_{start})$ that takes place when e_{start} occurs. Each time the physically-based simulation sends new meter values $m_i(t)$ to the signal conversion specification Ξ , the simulation time is increased, $t := t + \Delta t$, where Δt is the time increment of communication between the simulation and the control.

The specifications of human and artefact timing $\Xi_T = \{\zeta_{Ts}, \zeta_{Tl}, \zeta_{Tp}\}$ have been included because it is not possible to regulate *delaying* of transitions in logical constructs. The option of specifying delays can be useful to investigate the effects of hesitation or latency in human or artefactual information processing.

The specification of human or artefact timing is defined as

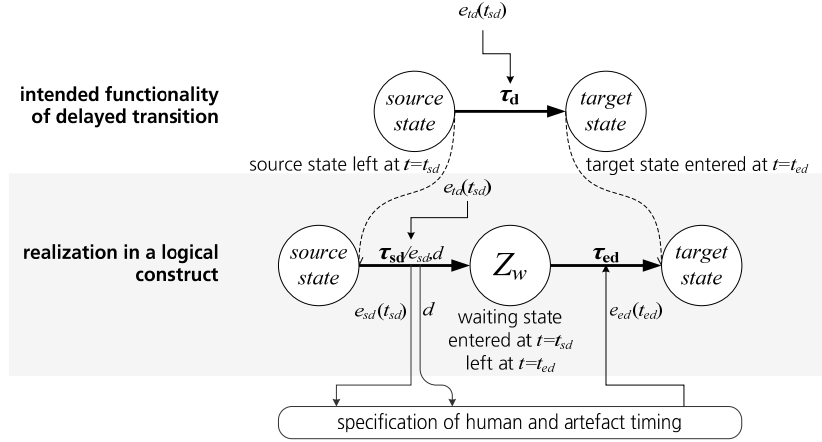
$\xi_{Tj} = \{\{e_{sd,i}\}, \{d_i\}, \{e_{ed,i}\}\}$, where

- $j = \{s, \ell, p\}$,
- $\{e_{sd,i}\} \supseteq \{e_{lgs,i}(\lambda_j)\}$ is the set of start-delay events to be received from the logical construct λ_j ,
- $\{d_i\} \supseteq \{u_i(\lambda_j)\}$ is the set of durations d_i of the delays (where $\{u_i(\lambda_j)\}$ is the set of logistic values produced by λ_j), and
- $\{e_{ed,i}\}$ is the set of end-delay events to be sent to λ_j .

A delayed transition $\tau_{d,i}$ in a logical construct λ_j is triggered at $t = t_{sd,i}$ by an event $e_{td,i}$ (*delay-triggering event*) but it takes place (i.e., λ_j assumes its target state) at $t = t_{sd,i} + d_i$. In λ_j , a delayed transition is specified as a compound of two regular transitions $\tau_{sd,i}$, $\tau_{ed,i}$ with a waiting state $Z_{w,i}$ in between:

$$\tau_{d,i} = \{\tau_{sd,i}, Z_{w,i}, \tau_{ed,i}\}$$

The start-delay transition $\tau_{sd,i}$ is in fact a dummy transition, while the end-delay transition $\tau_{ed,i}$ is the actual delayed transition (Figure 26). When at $t = t_{sd,i}$ a meter event takes place that triggers the start-delay transition $\tau_{sd,i}$, the logical construct assumes the waiting state $Z_{w,i}$. The start-delay transition $\tau_{sd,i}$ generates a logistic event called *start-delay event*, $e_{sd,i}$ which is sent to the timing specification ξ_{Tj} at $t = t_{sd,i}$, together with a *duration value* d_i . After ξ_{Tj} has received the



(The notation $\tau_{sd}/e_{sd},d$ is a common convention to specify "the transition τ_{sd} generates the logistic event e_{sd} and the logistic value d ")

Figure 26. Realization of a delayed transition in a logical construct.

start-delay event it generates the *end-delay event*, $e_{ed,i}$ at $t = t_{ed,i} = t_{sd,i} + d_i$ is generated which triggers the *end-delay transition* $\tau_{ed,i}$ out of the waiting state $Z_{w,i}$ in the logical construct λ_j .

3.7 Concluding remarks

In this chapter the theoretical elements that are needed to achieve the targeted system functionality have been purposefully derived. The elements of the theory have been deduced logically from the hypotheses that have been assumed to be true. The theory explains the principles underpinning (i) the simulation layer containing nucleus-based models of humans and artefacts, and (ii) the logistics layer containing the scenario bundle, which is a logical construct acting as a control mechanism over the simulation. This makes it possible to perform simulations of procedurally disjunct sequences of interactions based on the designer's conjecture of human decision-making.

To connect nucleus-based models and scenario bundles, and to include information processing by artefacts in simulations, two additional logical constructs and a signal conversion specification have been defined.

The theory presented in this chapter does not extend to the realization of software components for storing and processing constructs, models, and specifications. I have addressed these implementation issues in the next chapter. The elaboration in Chapter 4 also addresses the arrangement of, and data exchange between the software components in a way that unifies the processing scheme in Figure 21 and the reasoning model of human-artefact interaction in Figure 4.

4 PROTOTYPE IMPLEMENTATION

4.1 Objectives

As a proof-of-the concept implementation, a working prototype of the hypothesized new functionality has been realized based on the theory described in Chapter 3. The objectives were (i) to gain experiences with feasibility and computability in general, (ii) to demonstrate the functionality of the conceptualized system, and (iii) to allow validation of the theory based on a specific application case.

The objective of feasibility testing was to confirm that the theory can be converted into a structured set of processable algorithms and to demonstrate the usability. The goal of demonstrating the functionality was to show how the results are generated when the implemented theory is applied to the investigation of realistic use processes. The justification of the theory (Chapter 7) involved comparison with existing solutions, focusing on estimation of the required preparation effort and the time needed to run simulations.

The next subchapter, 4.2, is a specification and explanation of how the theoretical resources presented in Chapter 3 are converted into mutually connected functional processing units that have later on been realized as computer software. It has to be mentioned that in 4.2 detailing of the units and the connections has been kept independent of the implementation tools (i.e., programming language, configurable software components, and/or readily available software components). Subsequently, in 4.3, the approach for the proof-of-the concept implementation is elaborated, and, based on functional requirements, the actual software tools are selected. In 4.4-4.6, the activities required for creating physically-based models and control specifications with the selected software tools are described. Finally, Subchapter 4.7 wraps up this chapter with some concluding remarks and with an account of the connection to the next chapter.

4.2 Conversion of theoretical resources to the requested functionality

4.2.1 Operationalization of the conceptual model

The conceptual model for controlled simulations presented in Figure 21 has been decomposed into five principal constructs as was theoretically defined in 3.5 and 3.6. These are: (i) the scenario bundle, (ii) the model of low-level logical control of human motion, (iii) the procedure structure, (iv) the signal conversion specification, and (v) the physically-based simulation model. From an implementational

Table 5. Principal constructs of resource-integrated modelling and simulation

layer source of instantiation	logistic layer (execution of logical constructs)		physically-based simulation layer (simulation of physics)
designer's conjecture	scenario bundle	signal conversion specification	physically-based simulation model
designer's creational skills	procedure structure		
predefined	model of low-level logical control of human motion		

viewpoint, these concepts can be distinguished based on (i) their source of instantiation (i.e., whether they are instantiated by the designer based on conjecture or based on creational skills, or predefined) and (ii) the layer they reside in (i.e., the logistic layer or the physically based simulation layer) as shown in Table 5.

The logistics layer contains three logical constructs: (i) the *scenario bundle* that specifies the designer's conjecture about human decision-making during use, (ii) a *model of low-level logical control of human motion* built up from predefined chunks of knowledge about response selection and response execution in human muscle activation, and (iii) the *procedure structure* that controls motion of actuators in artefacts according to specifications of (software) designers. In addition to logical constructs the logistic layer also contains the *signal conversion specification*, by which the signal connections between the logical constructs and the physically-based simulation model are specified. It has two distinct sources of instantiation: (i) meter events for human control are specified based on conjecture, while (ii) meter events for control of the artefact are specified by designing.

The physically-based simulation layer consists of just one model, namely the physically-based simulation model, which is built up from nuclei. Although it is simulated as one model, it may have distinctive sub-models from different origins, namely for each artefact and for every human in the investigated use process, whereby the designed product is instantiated by the designer's creational skills. The other sub-models may be already available, e.g., from libraries. Beside the mechanical relations shown in Table 4 (3.4.4), the physically-based simulation model also contains relations describing low-level continuous control (e.g., PID control) of muscles and actuators.

4.2.2 General signal flows

Figure 27 shows the general signal flows of controlled interaction simulation as a result of combining the theory in Chapter 3 with its operationalization in 4.2.1. This figure maps the processing flows in Figure 21 (see 3.3) to the hypothesized reasoning model of human-product interaction in Figure 4 (see 1.6). The labels are explained in Table 6.

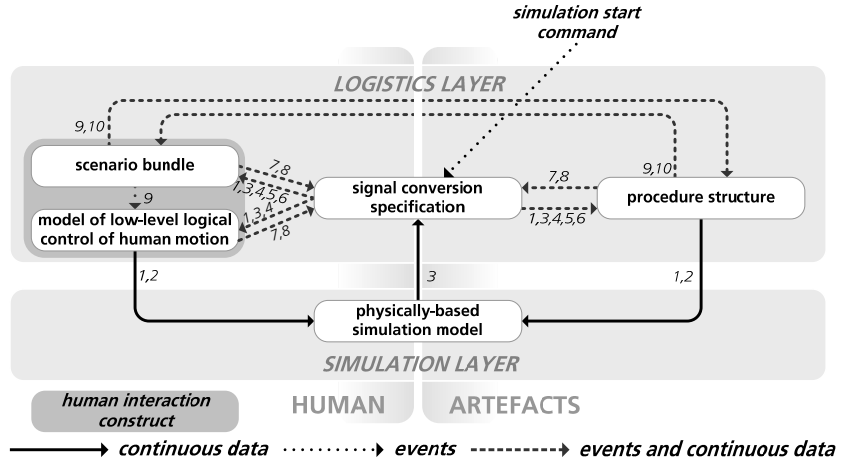


Figure 27. Signal flows of controlled interaction simulation

The generation of signals starts when the designer enters the simulation start command. At that time, the signal conversion specification sets the time to $t = t_0$ and it sends the start event to the scenario bundle and the procedure structure. The time signal is continuously transmitted through the logical constructs to the physically-based simulation. Further steps of processing are discussed in 4.2.7.

From the viewpoint of processing functionality, as well as inputs and outputs, the five principal constructs in scenario-bundle based simulation perform operations according to the following specifications:

The scenario bundle is a logical construct according to the formal definitions in 3.6.1. From the signal conversion specification it receives meter signals and events. It produces the following output: (i) logistic events to the model of low-level logical control of human motion and (ii) signals that control delays (hesitation, latency) as defined in 3.6.5.

The model of low-level logical control of human motion is the logical construct that generates the control value modifiers k_i to control human muscle activation. This model has been defined in 3.6.3. The scenario bundle and the model of low-level logical control of human motion together can be considered as the human interaction construct. This is one logical construct that realizes the 'complete' control process of generating control value modifiers based k_i on external events $e_{ext,i}$ (including meter events $e_{met,i}$), meter values m_i , and logistic values u_i (as it has been defined in 3.6.1).

The procedure structure is the artefactual equivalent of the human interac-

Table 6. Specification of the labels used in Figure 27, 28, and 40

1. time $t \in \{m_i\}$	6. end-delay events $\{e_{ed,i}\} \supseteq \{e_{met,i}\}$
2. control signals $\{p_i\}$	7. start-delay events $\{e_{sd,i}\} \supseteq \{e_{igs,i}\}$
3. mechanical meter signals $\{m_i\}$	8. duration data (for delays) $\{d_i\} \supseteq \{u_i\}$
4. meter events $\{e_{met,i}\}$	9. logistic events $\{e_{igs,i}\}$
5. start event $e_{start} \in \{e_{met,i}\}$	10. logistic values $\{u_j\}, u_j \notin \{d_i\}$

tion construct. Based on the same types of input and output (but a different selection of signals as specified by the designer), it generates signals to control actuators in simulated artefacts.

Finally, the physically-based simulation model simulates physical behaviour based on the time signal and on the control signals from the logistics layer. It produces meter signals that serve as input to the signal conversion specification. because of its central position in the processing scheme and the fact that it deals with all the signals in Table 6, except for control signals. The signal conversion specification is elaborated first in the next section.

4.2.3 Preparation of input signals for logical constructs: the signal conversion specification

Figure 28 shows the decomposition of the signal conversion specification into functional sub-specifications and the signals between them:

The *specification of stimulus recognition* for the human and for artefacts defines the meter events $e_{met,i}$ attached to transitions in the scenario bundle, the model of low-level logical control of human motion, and the procedure structure, respectively. Their specification includes orientations r_i , thresholds h_i , and event-generating functions $f_i(t)$. It also specifies which meter values $m_i(t)$ have been specified as condition values $v_i(t)$ to be evaluated with transitions in the respective logical constructs.

According to the above description, the conversions in the signal conversion specification can be considered as applying predefined ‘templates’ of operations to signals that have already been specified in logical constructs and in the physically-based simulation model. This means that the contents of the signal conversion specification can be automatically extracted and configured without explicit involvement of the designer.

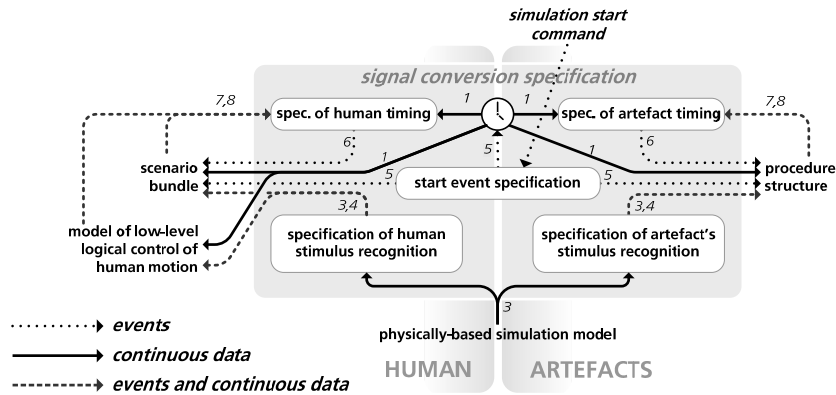


Figure 28. Decomposition of the signal conversion specification.

4.2.4 Specification and processing of human control of interaction based on scenario bundles

A scenario bundle is a logical construct by which designers specify interactions, operations, and behaviours in conceived use processes representing the related human decision-making. From a computational point of view, a scenario is a compilation of transitions τ_i corresponding to *decisions* that the user of the product is supposed to make in order to start or end particular *activities* specified as states Z_i .

The instructions specified in the scenario bundle correspond to human control that manifests itself on the level of decision making. They specify when to start or end high-level activities such as, for example, '*pull lever*', '*push button*', '*insert coin into slot*', '*turn left*', or '*open lid*' in conceived use processes. The instructions neither specify the body parts or muscles to be addressed, nor the quantitative values of control signals. The advantage of scenario bundle-based specification of interactions is that the designer does not have to put efforts in specifying low-level control.

In terms of signal flows this means that the scenario bundle sends outputs to the model of low-level logical control of human motion rather than generating output signals to the physically-based simulation model directly. The mentioned communication uses logistic events, which signify the points in time when high-level activities start or end. The activities themselves correspond to states Z_i in the scenario bundle, which carry the names of the activities. The logistic events transmitted to the model of low-level logical control of human motion are evoked by transitions τ_i between states. In the model of low-level logical control of human motion, transitions between low-level activities are triggered by these logistic events. In the scenario bundle, one high-level activity Z_i may correspond to a group of low-level activities $\{Z_i'\}$ interconnected by low-level transitions $\{\tau_i'\}$.

The waiting states $Z_{w,i}$ that were introduced in 4.2.3 also correspond to 'activities' in the sense as described above, with one distinction. By way of exception, logistic events evoked by transitions to and from waiting states are sent to the signal conversion specification rather than to the model of low-level logical control of human motion.

4.2.5 Specification and processing of low-level logical control of human motion

As explained above, the model of low-level logical control of human motion is a logical construct in which the states $\{Z_i'\}$ are low-level activities regarding motion control of specific body parts. The model of low-level logical control of human motion receives logistic events from the scenario bundle as its input, as well as meter signals and meter events from the signal conversion specification. Its outputs are control signals (and also the time signal) to the physically-based simulation model. As was defined in 3.6.3, each control signal $p_i(t)$ changes in time as an effect of taken transitions, which results in a signal transmitted to the

simulation that can be described as a step function.

The model of low-level logical control of human motion during interaction with a specific product is instantiated based on various chunks of knowledge from human motor science about human response selection and response execution. It is assumed that many activities in the scenario bundle can be recognized as common interaction patterns and that their variations can be parameterized. For these activities, the process of instantiation can be automated, e.g., 'reaching with the hand for a point (x, y, z) ', or 'pushing a button with stroke s and orientation (α, β, γ) '.

Together the model of low-level logical control of human motion and the scenario bundle are processed as a *human interaction construct*. A flowchart of the processing performed by this construct can be found in Appendix 3.

4.2.6 Specification and processing of embedded logical control in artefacts: the procedure structure

The *procedure structure* makes it possible to include the artefactual counterparts of perception, decision-making, and muscle activation as additions to human interactions in simulations. The procedure structure represents the instructions programmed into an artefact's control mechanisms according to the intents of designers. It has the same inputs and outputs as the human interaction construct⁴⁵, and it also processes signals according to the flowchart in Appendix 3.

4.2.7 General process flows of controlled interaction simulation

In Figure 29, the order of performing processing steps in time during controlled simulation is shown as a flowchart. It has been subdivided into processing steps performed by algorithms for (i) physically-based simulation, (ii) processing of the signal conversion specification, and (iii) execution of the logical constructs (i.e., the human interaction construct and procedure structure). Processing takes place in discrete time steps at a fixed interval Δt . The time steps are controlled by the processing of the signal conversion specification. Once the designer has given a start command, the signal conversion specification starts the time at $t = t_0$ and, for the first time step, it produces the start event $e_{start}(\uparrow, t_0, t)$ for a basic processing loop, as depicted by the arrows in Figure 29. Execution of the logical constructs starts with the one meter event, e_{start} , and the one meter value, $t = t_0$, that just became available. Consequently, the initial transition τ_0 takes place in each of the logical constructs⁴⁶. Logistic events $e_{lgs,i}(t_0)$ caused by these transitions may trigger further transitions τ_i , more consequential logistic events $e_{lgs,i}(t_0)$, and additional transitions, etc. Within the logical constructs, such chains of

⁴⁵ The procedure structure represents a 'design' of software, and it is up to the designer/programmer to structure it as needed. Hence, a prescribed separation into high-level control and low-level control as in the human interaction construct is not needed.

⁴⁶ Initial transitions are specified by connecting the initial state Z_0 to one or more target states.

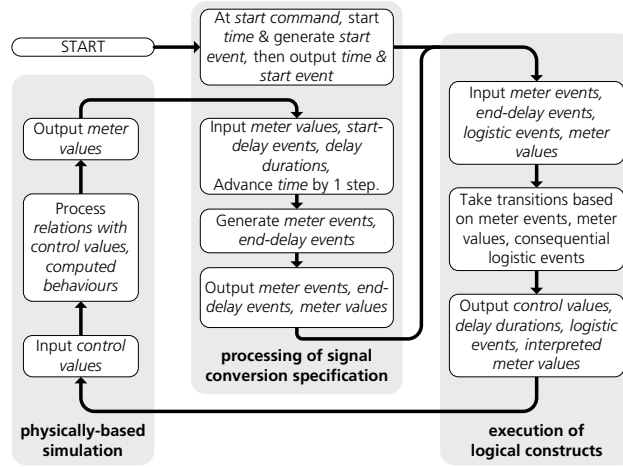


Figure 29. Flowchart representation of controlled simulation

transitions and logistic events are processed until no logistic event is available to trigger further transitions. Afterwards, if available, the following signals are exported from each logical construct: (i) control values $p_i(t_0)$ for the physically-based simulation model, (ii) delay durations d_i and (logistic) start-delay events $e_{sd,i}$ for the signal conversion specification, and (iii) logistic events $e_{lgs,i}$ and logistic values $m_{int,i}$ for the other logical construct. Actually, the latter signal represents the ‘direct information exchange’ in Figure 4 on page 9. The signals $p_i(t_0)$, d_i , and $e_{sd,i}$ are processed during the same time step, while $e_{lgs,i}$ and $m_{int,i}$ are to be processed during the next time step.

The subsequent processing steps are performed by the physically-based simulation layer. It imports the control values $p_i(t_0)$ just produced by the logical constructs. These values control the activation relations of the modelling elements representing muscles and actuators. If no new control values are available, then the previous ones are used. If no previous values are available, then defaulted ones are used. Based on these control values and predefined mathematical formulae within the physically-based simulation model, the mechanical behaviour of the whole human-artefact system is computed for the current time step. This affects meter values $m_j(t)$.

These meter values $m_j(t_i)$ $t = t_i = (t_0 + \Delta t)$, are then output to the signal conversion specification at the end of the time step. After increasing the time by Δt , the signal conversion specification uses $m_j(t_i)$ to generate events $e_{met,i}(t_i)$ according to what has been specified by the designer. The signal conversion specification also imports start-delay events $e_{sd,i}$ and delay durations d_i that have been generated by the logical constructs during the current time step.

An event $e_i(t) = 1$ is generated if, as a consequence of the latest changes in meter values, its specified threshold value is crossed with the specified orientation. Otherwise, all events have the defaulted value $e_i(t) = 0$. These processing rules apply both to meter events and to end-delay events (for which the time t is the meter value). To enable the execution of the instructions of the next time

step, the signal conversion specification sends the following signals to the logical constructs: (i) the generated events $e_{met,i}(t_i)$ and (ii) the meter values $m_j(t_i)$ that have been specified as condition values v_i .

Processing of a next time step $t_{i+1} = (t_i + \Delta t)$ starts with importing events and meter values to the logical constructs. The actual processing takes place in the same steps as described above. There is a difference, however, in the nature of the events received by the logical constructs. In the first time step only the start event was processed, while during the next steps three other types of events are imported for processing, namely (i) meter events $e_{met,i}(t_i)$ and end-delay events $e_{ed,i}(t_i)$ from the signal conversion specification, and (ii) logistic events $e_{lgs,i}(t_i)$ from the respective other logical construct.

The controlled simulation may be ended at any time by stopping the computational processes. This has not been included in the flowchart.

4.3 Selection of tools for a proof-of-concept implementation

4.3.1 Functional requirements

It was found that to transfer the theory to a fully functional software solution that allows multi-aspect investigation of use processes, software components with the following functionalities are needed:

- (i) A repository of nucleus-based models of a variety of human bodies
- (ii) A nucleus-based artefact modelling system, or a system that supports the conversion of other CAD models to nucleus-based models and that allows insertion of models from (i);
- (iii) A repository of logical models of low-level human control behaviour tailored for controlling the models of (i);
- (iv) A subsystem for the specification of logical constructs (in particular, scenario bundles and procedure structures) that allows insertion of models from (iii);
- (v) A subsystem that generates a signal conversion specification, and connects input/output signals of the nucleus-based model and logical constructs;
- (vi) A subsystem for nucleus-based mechanics simulation of the models created with (or produced by) (ii);
- (vii) A subsystem for execution of the logical constructs specified with (iv).
- (viii) A subsystem for concurrent processing of (vii) and (viii), providing output to designers in the form of animated motion of the simulation model and numerical values of variables over time, as well as a visualization of the 'path' taken through the scenario bundle.

The proposed theory underpins the functionalities realized by components (iv), (v), (vii), and (viii). According to my best knowledge, no software tool or package that is able to provide this combined functionality is available. Consequently, the resources to make the proof-of-concept implementation possible had to be developed and/or adapted.

In their functioning, these components depend on the other components: concurrent processing (viii) needs nucleus-based simulations, and a scenario bundle that is connected to a model of low-level human control (iii), which in turn need nucleus-based models created with (ii), and human-body models from (i). The forerunning survey pointed out that no commercial software is available to realize these other functionalities⁴⁷. Nevertheless, they needed to be implemented and adapted into the proof-of-concept prototype in some way. It should be mentioned as a fundamental consideration that fully featured implementation has not been a goal of this PhD project but that the feasibility of realizing the functionalities realized by the abovementioned components (iv), (v), (vii), and (viii) had to be proven. Therefore it has been decided to realize the proof-of-concept prototype with existing adaptable tools wherever that was possible.

4.3.2 Realization approach

Considering the facts discussed above, I took the following approach to realize the functionality of the proof-of-concept prototype.

To realize (i) specification of logical constructs, (ii), generation of the signal conversion specification, (iii) execution of logical constructs, and (iv) concurrent processing during simulations, I have used adaptable tools to create simulatable/executable models and constructs and according to theory. Minor deviations have been translated to the theoretical concepts in a well-documented and unambiguous way.

Since the *user interface for the creation* of constructs is not addressed in the theory, I used the user interfaces offered by the selected tools (and which will be specified in 4.3.5 and 4.3.6.).

Nucleus-based human and artefact models and the model of low-level logical control of human motion have been realized as *proxies* by using existing systems with which they could be modelled and that could be used to simulate their behaviours. For these models, the priority has been that they exchange signals with the logistics layer as has been specified in the theory. Full conformity of the contents of the proxy nucleus models to the nucleus theory and of the proxy model of low level control of human motion to a validated control model based on human motor studies has been not been my objective. For the nucleus models, the conformity was kept limited to the principal concepts, i.e., particles and relations.

4.3.3 Simulation and modelling for simulation

The following criteria have been considered at selecting the proxies for nucleus-based modelling and mechanics simulation:

- (i) resemblance of the concepts that a proxy is based on and the concepts

⁴⁷ The mentioned components mainly address nucleus-based modelling and simulation, which is a current research topic of the CADE section. Future plans include dedicated software development.

manifested in nucleus-based models. In particular, it should be possible to create models in which the unary and binary mechanical relations in Table 4 (p. 63) can be recognized;

- (ii) the possibility to control simulations with values obtained from logic processing;
- (iii) the possibility of defining behavioural models using imported 3D CAD models, and to edit and manipulate models in a 3D environment;
- (iv) the possibility to visualize simulation results by animated motion of 3D human/artefact models.

The investigation discussed in Chapter 2 showed that two groups of commercialized systems for modelling and simulation of mechanical behaviour are commonly used by the industry, namely FEM systems and multibody dynamics systems. Criterion (i) is best fulfilled by the capabilities of multibody dynamics systems. Unlike FEM systems, they allow the specification of the relations in Table 4 based on similar concepts. Although FEM systems are typically considered to be better at modelling and simulation of large deformations of continua (i.e., the behaviour inside a component), this functionality is not realized with binary relations between particles, as in nucleus-based models⁴⁸. This is why I decided to use a commercialized multibody dynamics package for the proof-of-concept implementation of nucleus-based modelling and simulation.

To this end, the next step has been to compare commercialized multibody dynamics systems based on criteria (ii)-(iv). According to Chang et al. [258], two groups of multibody simulation packages dominate the market: one of them is based on Adams (Automatic Dynamic Analysis of Mechanical Systems) and the other is based on DADS (Dynamics Analysis and Design System). As a representative of the Adams-based systems I investigated MSC Adams, and from the DADS family I investigated LMS Virtual.Lab. As representatives of other approaches, I also involved simpack⁴⁹, and CAMELview⁵⁰ in the comparison. The information regarding the capabilities of these systems was obtained from the original software vendors and from related literature.

In terms of controlling simulations with values provided by logic processing, each of the investigated systems allowed external control through linking with a particular third-party software package, namely, Matlab Simulink. It was observed that an 'internal' graphical representation for logical control triggered by changes in (meter) values was available in systems that were commercialized at the time of beginning the PhD research. Nevertheless, systems are developing in this direction, which is evidenced by a planned extension of CAMELview. In principle, conditional change of control values triggered by changes in other (meter) values can also be done by using textual logical descriptions (using IF-THEN state-

⁴⁸ ... or in discrete-flexibility models (see 2.3.8), which can actually be realized using multibody systems.

⁴⁹ from INTEC, www.simpack.com

⁵⁰ from ixtronics, www.ixtronics.com

Table 7. Comparison of commercial multibody simulation software packages

Software		MSC Adams	LMS Virtual.Lab Motion	CAMELview	SIMPACK
Criterion					
Logical control of simulations	Interfacing with third-party software	yes (with Matlab / Simulink)	yes (with Matlab / Simulink)	yes (with Matlab / Simulink)	yes (with Matlab / Simulink)
	Using a graphical notation	no	no	no (planned for a next release)	no
	Using textual descriptions: IF-THEN statements, etc.	yes	yes	yes	no
	Using algebraic descriptions: (approximated) Heaviside step functions	yes	yes	yes	yes
3D model management capabilities	3D CAD import possible	yes	yes	yes	yes
	3D modelling interface	yes	yes	no	yes
Graphical visualization and animation		yes	yes	yes	yes

ments) or algebraic descriptions, for instance by (approximated) Heaviside step functions [259]. To define consecutive transitions of the same variable at multiple points in time, the statements or functions must be combined, e.g., by nesting. All of the investigated systems support these options of specification of logical control with the exception of SIMPACK, which does not support text-based logic.

As far as 3D modelling is concerned, all software packages allow import of 3D CAD models. With the exception of CAMELview, all of the investigated packages also have their own graphical 3D modelling interface. In CAMELview, the connections between parts are specified in 2D block diagram elements (similar to SimMechanics, see 2.3.4) and 3D models of individual parts must be accessed through the blocks that represent them. 3D model editing can only be done via the keyboard, using a dialog in which one part is visible at a time. All of the packages can display simulation results as 3D animations.

Table 7 summarizes the comparison of the investigated commercial systems. The two most widely applied systems, Adams and Virtual.Lab, offer the best match to my criteria, while the other competitors are closely behind.

In addition to the above facts, a reason to prefer Adams over Virtual.Lab is that Adams models can be imported into Virtual.Lab, but not vice versa [110]. This means that once created, Adams models still offer the opportunity to transfer them to the Virtual.Lab environment, for instance when the Adams proxy turns out to be inadequate for unforeseen reasons. Consequentially, taking into account all the above considerations, MSC Adams has been selected as the proxy implementation tool for the conceptualized nucleus-based modelling and simulation approach.

4.3.4 Human body modelling

The separately available software package LifeModeler (see 2.3.6) offers a wide range of functions for human body modelling that can be used in MSC Adams simulation models. However, rather than opting for a comprehensive human body model I decided to apply various simplifications in the proxy models that I created of the human body. Firstly, I did not exploit those elements of intended

system functionality that enable instantiation of human bodies from a library. This would have made it possible to instantiate multiple users and to investigate of use processes with different users. However, since one instance of a human body model suffices for the goals described in 4.1, I did not take this opportunity that would have been possible by using LifeModeler.

Secondly, the human body was simplified in two respects. Muscles have been modelled as linear actuators acting between points on two body parts. This corresponds to the situation in a real human body. However, instead of two contracting muscles acting across a joint on two sides, only one 'muscle', which is capable of both contracting and expanding, has been modelled. The other anatomical simplification is that the phalanges of fingers are rotated by surrogate muscles between them rather than by tendons that are connected to muscles in the forearm.

Thirdly, to describe the various mechanical properties and relationships of the human body, a trial-and-error strategy was applied. This involved varying coefficients of friction, damping, stiffness, etc., and varying connection arrangements between particles, until a model was obtained that behaved sufficiently reliable during simulations. Furthermore, the springs and dampers that have been used to model deformation behaviour have been specified as linear.

Fourthly, surrogate particle clouds were used only for those parts of the model in which deformations were expected as an outcome of the interaction in the simulated use process. All other parts in the model were treated as non-deformable objects.

Finally, as was already mentioned in 4.3.2, in order to shorten modelling and simulation time the resolution of particle clouds was kept relatively low. It means that only dozens of particles have been used to instantiate the complete simulation model. However, for absolutely accurate modelling thousands or even million particles would be needed.

4.3.5 Specification and execution of logical constructs

Considering their theoretical definitions, the logical constructs introduced in 3.6.1 resemble existing representations of finite automata. This gave the basis of my decision on using commercially available finite automata software to realize the targeted proof-of-concept implementation as a proxy for the specification and execution of logical constructs.

For adequate specification and execution of logical constructs, the selected software (and the automata representation on which it is based) must fulfil the following requirements:

- (i) The software must allow its user to maintain distinct constructs for (a) the scenario structure, (b) the models of low-level logical human control, and (c) the procedure structure. The software should enable control of a single simulation model through concurrent execution of these constructs.
- (ii) Based on the definitions in 3.6, the software must allow (a) interfacing with simulations to exchange meter signals m_i and control signals c_i , and (b) specification of events e_i , transitions τ_i , states Z_i , and transition conditions γ_i .

- (iii) The software must support specification and modification of logical constructs by a graphical representation
 - (iv) The graphical representation should make it possible for user to monitor (or review) the succession of transitions and states during (or after) a simulation
- Requirement (i) addresses the representation potential. Concretely, it means that the graphical representation that is used should support concurrency and hierarchy. According to the discussion in 2.5.2 concerning the graphical representations for finite automata, statecharts and Petri nets are the most widely used in practice and they offer the required representation potential. This reason narrowed down my search for software for specification and execution of logical constructs to these two representation forms.

Requirement (ii) addresses interfacing with the physics simulated by Adams, which is capable of exchanging control-related signals with Matlab Simulink and with MSC Easy5. According to information from MSC, Easy5 does not provide means to specify logical control based on statecharts or Petri nets. Simulink offers a Stateflow toolbox, which enables specification of logical constructs using a statechart dialect [260]. This toolbox is widely used in the industrial practice [261]. Simulink's Stateflow toolbox fulfils the above requirements with two restrictions, namely, that (ii) specification of events must be done in Simulink outside the Stateflow toolbox, and (iv) successions of transitions and states can be monitored only during simulations, and not afterwards.

Alternatively, the same functionality can be provided by Petri-net software that allows linking of its logical constructs to Simulink. In particular the Netlab [262] software tool is appropriate for this purpose but it is a university-developed tool rather than accepted standard industry software. Another alternative is to convert logical constructs to soft prototypes for PLDs (programmable logic devices). This can be done with Petri nets [263] or statecharts [264]. Such a soft prototype is a block representing all processing actions of the logical construct. It can be inserted in a Simulink block diagram. The disadvantage of this approach is that the soft-prototyping blocks are actually 'black boxes'. It means that monitoring or reviewing the path of transitions and states taken through the logical construct is not possible.

Considering the evaluation of the abovementioned options based on my requirements, I have selected Simulink Stateflow for specification and execution of logical constructs because unlike the other investigated options it fulfils all my requirements. By using Stateflow in prototyping of the logistics layer, I could realize the proof-of-concept prototype by using only two software packages (Simulink and Adams), which is an additional advantage. A further possible advantage of Stateflow is that it can also be used to specify and execute Petri nets [265,266]. This might offer a convenient fall-back arrangement it turns out that the representation potential of Statechart is insufficient.

4.3.6 Models of low-level logical control of human motion

In the pilot implementation, the repository of logical models describing low-level human control behaviour has been realized by a proxy rather than by developing an implementation based on validated models. At the time of starting the PhD

research, no such models were known to be available and developing them was not considered feasible with the resources available for this PhD research project. However, in the meantime new developments have indicated that simulation of low-level control based on validated models is possible (see 2.5.1).

The proxy construct of low-level control of human motions had to bridge the gap between the scenario bundle and the simulation, i.e., it has to convert the logistic events the scenario bundle produces based on descriptions of human decisions conjectured by the designer to quantitative control values for the simulation. To resolve this, I adopted an approach often taken in programming robotic human-body imitations. Programmed instructions in this application area are typically logic-based and aimed at obtaining successful interaction *results* rather than at realistic motions *during* the interaction⁵¹. For instance, for the Utah/MIT dextrous hand [267] it was more important to obtain grip on an object so that it could be carried, than to achieve realism in the preceding motions of the fingers and the arm.

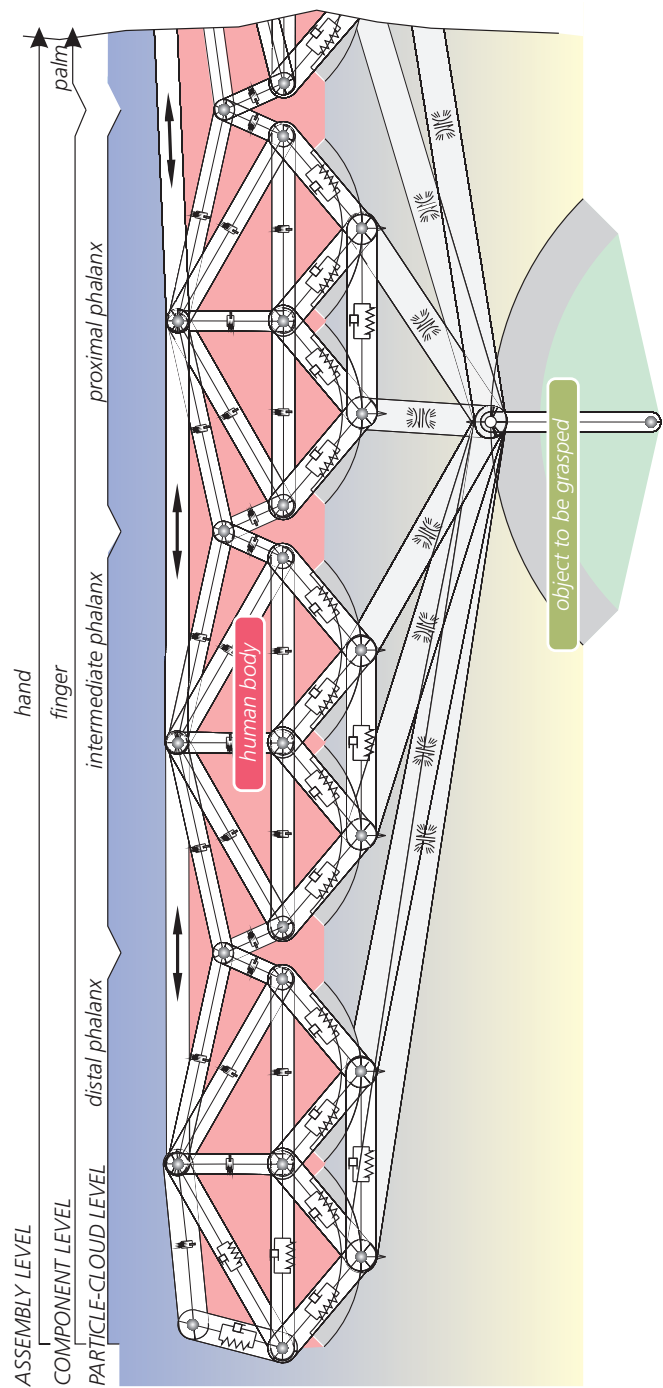
In the proxy model of low-level logical control of human motion I have 'programmed' instructions for moving the fingertip from A to B. I did not care whether the used motion patterns were natural and efficient. In a future implementation, these workarounds are to be replaced by sophisticated instructions based on invariants (see 2.4.3) and/or findings from other research [e.g., 191,193].

4.4 Building proxies of nucleus-based simulation models with Adams

4.4.1 Introduction

In the proof-of-concept implementation, physically-based simulation models have been created using the basic solid-modelling instructions offered by Adams. As an example, Figure 30 shows how relations-based modelling with nuclei has been applied to a representative part of a human-artefact system. Actually, this is a simplified 2D model of the human finger. The nuclei that have been used in this model have been instantiated for the relations listed in Table 4 on page 63. Below I will elaborate on how proxies of the specific nucleus-based modelling elements (entities on five levels according to 3.4.3, and relations according to 3.4.4) have been realized.

⁵¹ Actually, exchange of algorithms between the fields of robotics and human motor control appears to be common practice [171].



(legend: see Table 4 on page 63)

Figure 30. Low-resolution nucleus-based model of a human finger and an object to interact with, shown as a simplified 2D representation

4.4.2 Instantiation of entities at the five levels of modelling

Nucleus-based modelling distinguishes modelling entities on five levels, namely, particle, particle cloud, component, assembly, and system (3.4.3).

Starting at the lowest level, a proxy of a particle π is generally instantiated using the point_mass entity available in Adams:

```
part create point_mass name_and_position &
  point_mass_name = .model_1.particle_pi &
  Adams_id = 8&
  location = 0.0, -1.4E-002, 1.0E-002&
  orientation = 0.0d, 0.0d, 0.0d
```

Particles are connected by relations as defined in 3.4.4 (which will be elaborated specifically for the pilot implementation in 4.4.3) in order to instantiate particle clouds. The boundary particles that represent the bounding surfaces of a particle cloud need special attention because, (i) the point mass entity in Adams cannot be used to specify them and (ii) they need a reference point on the surface of the object represented by the particle cloud. Both issues are related to the requirement that to each boundary particle an infinitesimal half space must be assigned.

The first issue is explained and elaborated as follows. In Adams, oriented surfaces can be assigned to objects, which can in principle be used to instantiate half spaces. However, this is not possible if the object is a point mass, since point masses in Adams have no orientation that can serve as a reference to align the surface. The only available alternative for the instantiation of boundary particles is to instantiate them as volumetric objects.

The second issue is the fact that for a boundary particle it is required that its reference point lies on the surface of the object represented by the particle cloud. If the particle is instantiated as a volumetric object, this can be resolved if (part of) the surface of that object acts as a proxy of the required half space, and if the reference point of the particle is on that (part of) the surface. However, this solution poses a problem for boundary particles of bodies that collide with other entities. This applies for instance to the particles representing the skin on the human finger in Figure 30.

Figure 31 illustrates the problem that any contact normal force acting on the proxy half-space at an offset from the reference point of the boundary particle will cause the proxy half-space to rotate around its reference point. This is a rotation that cannot be observed in the real world, but in the model it can be prevented only if the orientation of the half-space is maintained by a reaction torque. Such a torque can exist only if it is caused by relations of the particle with other particles, i.e., spring-dampers. If the spring-dampers are all connected to the reference point on the colliding surface, this is not possible.

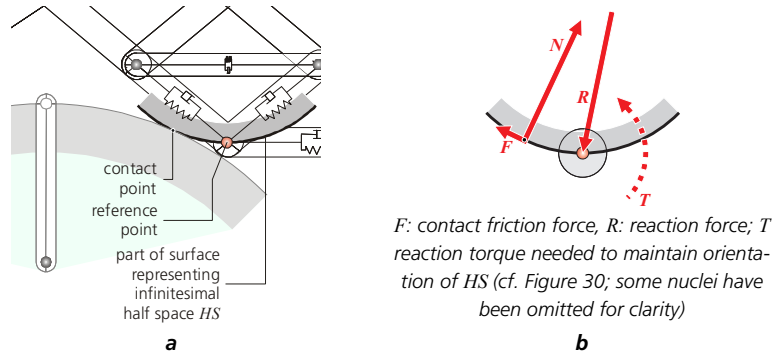


Figure 31. Contact normal force N acting on an infinitesimal half-space HS at an offset from the reference point of a boundary particle. a. overview; b. free-body diagram

In my model of the human finger I worked around this issue by instantiating each boundary particle as a small sphere (0.5mm radius) of which the surface acts as a proxy half-space, and of which the reference point is at its centre (Figure 32). This configuration eliminates the offset, and thus the need for a relation causing a reaction torque. The proxy does not conform to the conventions of nucleus-based modelling, which prescribe that the reference point of a boundary particle must be a point on the surface. Moreover, the workaround is only valid if the entity to be grasped is sufficiently large and has no surface details that can penetrate the gaps between the small spherical surface patches that act as proxies of half-spaces. In fact, this is the case if the boundary particles of the entity to be grasped have been instantiated using the same workaround.

It is for that reason that, in the pilot implementation, the following additional restriction was applied in modelling colliding pairs of entities. At least one of the colliding entities was *not* instantiated as a discretized particle cloud with a mesh of boundary particles at its surface. Instead, it was instantiated as a simplified particle cloud consisting of a centre-of-mass particle, connected with a limited set of boundary particles. To each of these boundary particles, a finite half-space was attached, together forming a closed surface around the centre of mass. (For instance, for an entity representing a prismatic cylindrical object, the centre-of-mass particle was connected to three boundary particles. To one of

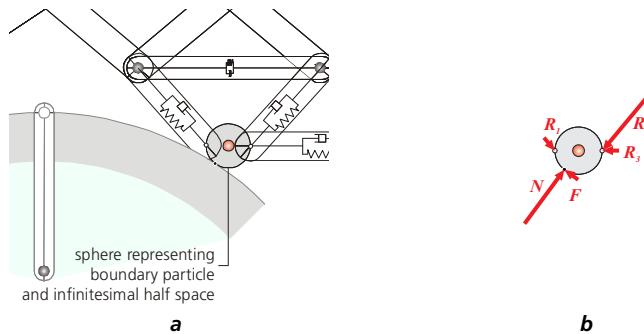


Figure 32. Sphere acting as half-space to eliminate the offset shown in Figure 31. a. overview; b. free-body diagram

these, a cylindrical half-space was attached, and to each of the other two, a planar half-space was attached.) This way, entities with closed surfaces could be instantiated that cannot penetrate each other or other objects with surfaces discretized into meshes of boundary particles.

An important drawback of this restriction is that the pilot implementation did not allow investigation of collisions between two flexible bodies (for instance, a hand squeezing a toothpaste tube). After all, a flexible body requires a flexible surface, which means that it must be discretized into a mesh of boundary particles connected to each other, and to the internal particles, by spring-damper relations.

The modelling levels above particle level are conglomerates of entities of each previous level. For the assignment of entities to the levels of particle cloud, component, assembly, and system the *group* command in Adams is used, for instance

```
group create &
  group_name = .model_1.pcloud_phalanx3 &
```

to instantiate a particle cloud of selected entities belonging to the third phalanx of the index finger. Names of groups must be entered manually, and they should include a reference to the level, e.g., *pcloud_name*, *comp_name*, *assy_name*, *sys_name*.

4.4.3 Instantiation of relations

To instantiate relations that are specified in nuclei, I have used relations that were available as standard building blocks in Adams. Of the relations defined in Table 4, some had to be disregarded, while others have been considered either implicitly or explicitly.

I had to disregard the *existence*, *reference*, and *connectivity* relations, because these relations are specifically needed for the internal data structure of dedicated nucleus-based models. Adams has its own data structure that is closely linked to its simulation algorithms. In order to maintain simulatable models, I have not considered modifying the internal data structure of Adams.

The relations (i) *surface normal vector*, (ii) *positioning*, (iii) *differential description of half space*, (iv) *distance*, and (v) *facing* have been considered implicitly. These relations concern the geometric specification of entities and their placement with respect to each other. While instantiating entities using the graphical modelling interface, Adams instantiates relations that establish surface normal vectors (i.e., distinction between inside and outside of objects), positioning, surfaces (which serve as proxies for half spaces, see 4.4.2), distances, and mutual orientations of pairs of surfaces (facing). The relations in Adams do not correspond to the nucleus-based relations on a one-to-one basis. Making them correspond explicitly was not considered because it would mean that I had to modify the internal data structure of Adams.

The remaining relations, *mass*, *prescribed (angular) velocity*, *friction*, *collision*, *prescribed force/torque*, and *spring-damper* have been instantiated explicitly in Adams. These physical behavioural relations play an important role in simu-

lations. Moreover, the prescribed velocity/force/torque relations are used for the specification of control of muscles and actuators.

A mass relation is instantiated by assigning it to a particle (in this case `particle_pi`) as follows:

```
part modify rigid mass_properties &
    part_name = .model_1.particle_pi &
    mass = 2.0 &
```

It should be noted that in Adams terminology mass is considered a 'property,' and a particle, like any metric object, is considered a 'part'.

The *prescribed (angular) velocity* kinematical relation is instantiated indirectly through a kinetic relation, namely a *prescribed force* or *prescribed torque*. It is for that reason that the implementation of these kinetic relations is described first.

The kinetic *prescribed force relations* ϕ^{Fp}_{ij} , which correspond to muscles and actuators, exert forces or torques between particles. (From here, I will refer to these as *forces*. With a few changes, all that is said about forces can also be applied to torques). In the pilot implementation I have only considered forces and torques working on the lines through their two reference points o_i and o_j , so that only the magnitude had to be specified in the force relation $\phi^{Fp}_{ij} = \{o_i, o_j, ||Fp_{ij}||\}$ (see also 3.4.4).

I used the single-component force or `SFORCE` relation in Adams to instantiate a prescribed force between two points. Magnitudes of forces, specified as control values, are imported into Adams as so-called state variables.

For *prescribed (angular) velocities* I have only considered (angular) velocities in the direction of the line through reference points o_i and o_j , so that again only the magnitude had to be specified in the relations $\phi^{vp}_{ij} = \{o_i, o_j, ||vp_{ij}||\}$ and $\phi^{op}_{ij} = \{o_i, o_j, ||op_{ij}||\}$. Prescribed velocities cannot be imported as state variables, because the multibody simulation algorithms in Adams are based on forward dynamics. Therefore, prescribed velocities and prescribed angular velocities have been translated to forces and torques that control muscles and actuators. This is done by defining `SFORCE` relations based on feedback loops with proportional-integral controllers. In the case of a muscle force, where the velocity is an angular velocity around a joint:

$$F_{ij}(t) = K_P \{ ||\omega p_{ij}(t)|| - \omega_{actual}(t) \} + K_I \int_0^t \{ ||\omega p_{ij}(t)|| - \omega_{actual}(t) \} dt$$

with F_{ij} the `SFORCE` between reference points i and j (called 'markers' in Adams), and ω_{actual} the angular velocity caused by F_{ij} . The constants K_P and K_I (proportional and integral gain) determine the responsiveness and stability of the controller behaviour. In Adams, the abovementioned prescribed angular velocity is modelled for a linear actuator (e.g., a muscle) as:

```
part create equation differential_equation &
    differential_equation_name = omega_error_integral_i &
    Adams_id = 1 &
    initial_condition = 0.0 &
```

```

function = "omega_prescribed_i-omega_i" &
implicit = off &
static_hold = off

force create direct single_component_force &
single_component_force_name = muscle_i &
Adams_id = 103 &
type_of_freedom = translational &
i_marker_name = particle_i.MARKER_i &
j_marker_name = particle_j.MARKER_j &
action_only = off &
function = "propgain * (omega_prescribed_i-omega_i) +
intgain * DIF(omega_error_integral_i)"

```

or, for a rotational actuator:

```

[...]
type_of_freedom = rotational &
[...]

```

In Adams, the physical phenomenon relation describing *friction* and the kinetic relation describing *collision* have been combined in the so-called *contact* relation. This relation must be specified between surfaces of entities rather than between entities. For instance, a combined collision-and-friction relation between a particle *pi* that has surface *surface_i* and an object with surface *surface_j* is specified as:

```

contact create & contact_name = .model_1.CONTACT_1 &
Adams_id = 1 & i_geometry_name =
.model_1.part_particle_pi.SURFACE_i & j_geometry_name =
.model_1.part_object.SURFACE_j & stiffness = 10.0 & damping =
1.0E+008 &
exponent = 2.2 & dmax = 1.0E-003 &
coulomb_friction = on & mu_static = 2.0
& mu_dynamic = 1.2
& stiction_transition_velocity = 2.0
& friction_transition_velocity = 5.0

```

Finally, a (linear) spring-damper between reference point *i* on particle *b* and point *j* on particle *d*, both of which belong to a bone in the human finger, with spring stiffness *bone_stiffness* and damping coefficient *bone_damping*, is specified as follows:

```

ude create instance & instance_name = .model_1.spring_bone_bd &
definition_name = .MDI.Forces.spring & location = 0.0, 0.0,
0.0 & orientation = 0.0, 0.0, 0.0

variable modify & variable_name =
.model_1.spring_bone_bd.i_marker & object_value =
(.model_1.particle_b.MARKER_i)!

variable modify & variable_name =
.model_1.spring_bone_bd.j_marker & object_value =
(.model_1.particle_d.MARKER_j)!

variable modify & variable_name =
.model_1.spring_bone_bd.stiffness_mode & string_value = "lin-
ear"!

variable modify & variable_name =

```

```

.model_1.spring_bone_bd.stiffness_coefficient & real_value =
(.model_1.bone_stiffness)!

variable modify & variable_name =
.model_1.spring_bone_bd.damping_mode & string_value = "lin-
ear"!

variable modify & variable_name =
.model_1.spring_bone_bd.damping_coefficient & real_value =
(.model_1.bone_damping)!
    
```

To conclude about the implementation of nucleus-based modelling and simulation, it can be said that the proxy suffices for the definition and specification of those typical elements of nucleus-based modelling that are crucial for running simulations with models containing the mechanical relations given in Table 4. Using the Adams/control extension, meter signals are defined and the model created with the proxy is converted to a Simulink block (Figure 33) that can be inserted in a Simulink diagram, where it is connected to the logical constructs and the signal conversion specification. As a parameter setting of this `Adams_sub` block, the time increment Δt of the communication between simulation and control is specified as 'communication interval'.

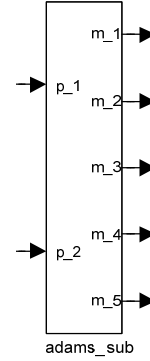


Figure 33. Example of a Simulink block representing a physically-based simulation, with two control signals as input and five meter signals as output

4.5 Specifying surrogate nucleus-based control instructions with Simulink Stateflow

4.5.1 Introduction

In the proof-of-concept implementation, logical constructs have been instantiated as Stateflow charts using the graphical interface offered by Simulink Stateflow. Stateflow charts form a dialect of the original statecharts developed by Harel [208]. The dialect is different in two respects: (i) it offers various enhancements in the form of building blocks for logical processing, and (ii) the graphical representation of parallel states is slightly different [260]. At the left-hand side, Figure 34 shows examples of the graphical appearance of Stateflow charts including all the basic specification elements that were used in the proof-of-concept implementation⁵². At the top right Figure 34 shows the equivalent of

⁵² only two of the Stateflow enhancements were used in the proof-of-concept implementation: (i) connective junctions and (ii) subcharting to create hierarchy with multiple levels.

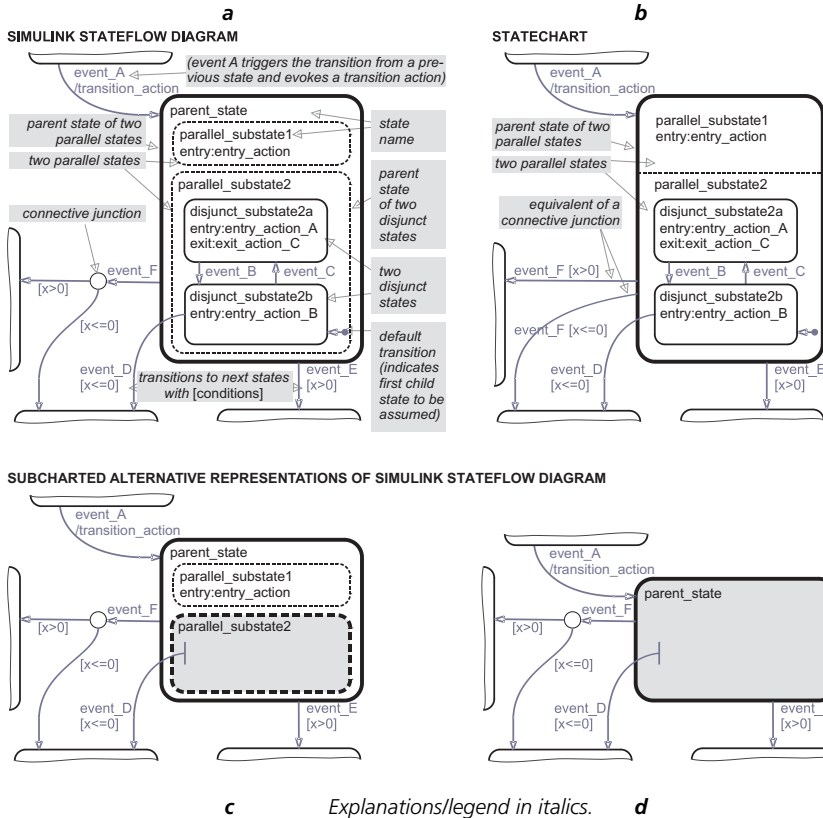


Figure 34. a: Excerpt from an example Stateflow chart with its graphical specification elements used in the proof-of-concept implementation. b: state-chart equivalent of the Stateflow chart. c and d: alternative representations of Stateflow chart based on subcharting.

the same diagram as a regular statechart. To facilitate hierarchical decomposition, Stateflow allows *subcharting*. This means that connected groups of states and transitions are demoted to subcharts, the contents of which are hidden from the higher-level view.

The further elaboration of logical constructs is organized as follows. First, the proof-of-concept implementation of the basic specification elements is elaborated in 4.5.2. by mapping the Stateflow elements shown in Figure 34 to the elements defined in 3.6 and 3.6.1. After that, specific implementation aspects of the three different logical constructs (i.e., scenario bundle, model of low-level logical control of human motion, and procedure structure) are elaborated in 4.5.3.

4.5.2 Implementation of basic specification elements

In accordance with the definitions in 3.6 and 3.6.1, the following basic specification elements have been implemented using Stateflow charts: (i) states and transitions with transition conditions, (ii) logistic events and (iii) change relations.

Referring to Figure 34, this has been done as follows:

Just like in the logical construct in Figure 23, on page 68, states Z_i and transitions τ_i in Stateflow charts are specified as nodes and vertices. As Figure 34 shows, parallel states are specified as child states of a parent state, which is different from the arrangement in Figure 23. To each state Z_i , a name is assigned. After the name, *actions* can be specified. In logical constructs, actions (specified as `entry:action` or `exit:action`) are used to specify the generation of logistic events and to instantiate change relations, as will be explained below. A triggering event $e_i(t)$ can be assigned to a transition. This can be a meter event imported from the signal conversion specification Ξ , or an internal or external logistic event. If no event is specified, any event occurring when the transition is enabled, $\varepsilon(\tau_i, t) = 1$, will trigger it. Actions to specify the generation of logistic events and/or to instantiate change relations can also be assigned to transitions. This is specified as `/action`. Finally, transition conditions $\gamma_i = g_{i\alpha}((v_1(t), \dots, v_n(t))) \diamond c_{i\alpha}$ are specified as `[g_i \diamond c_i]`, e.g., `[x+y > 4]`.

A logistic event $e_{lgs,i}(t)$ is used to synchronize transitions within one logical construct or across multiple logical constructs. There are two types of logistic events: explicit logistic events and implicit logistic events. Explicit events are explicitly specified in the Stateflow chart as actions. This is done either by assigning an action `/expl_event` to a transition τ_i or by assigning an event `entry:expl_event` or `exit:expl_event` to a state Z_i . Table 8 shows how these different explicit events can be used to synchronize a transition τ_j with one or more other transitions.

An implicit logistic event is an event that is automatically generated and named by Stateflow when a state becomes active or inactive. The implicit logistic event is called `enter[state_name]` or `exit[state_name]`, respectively. Table 8 shows how implicit events are used to synchronize a transition τ_j .

A change relation $\delta_i(\tau)$ of a transition τ specifies the changes in a set of control values $\{p_{\delta i}\}$, which are all parameters of one nucleus N . When τ takes place

Table 8. Synchronization of transitions through explicit and implicit events.
For the graphical realization see Figure 34.

explicit events	IF the action	<code>/expl_event</code>	is assigned to a transition τ_i	and <code>expl_event</code> is the triggering event of τ_j , THEN τ_j takes place synchronously with	τ_i
		<code>entry:expl_event</code> or <code>en:expl_event</code>	is assigned to a state Z_i		any transition of which Z_i is the target state
		<code>exit:expl_event</code> or <code>ex:expl_event</code>			any transition of which Z_i is the source state
implicit events	IF	<code>enter[Z_i]</code> or <code>en[Z_i]</code>	is the triggering event of τ_j , THEN τ_j takes place synchronously with		any transition of which Z_i is the target state
		<code>exit[Z_i]</code> or <code>ex[Z_i]</code>			any transition of which Z_i is the source state

the changes assign new values to $p_{\delta i}$ prescribed by the modifiers k_j :

$$\begin{aligned} p_{\delta 1}(\tau) &:= k_1 \\ &\vdots \\ p_{\delta j}(\tau) &:= k_j \\ &\vdots \\ p_{\delta n_{\delta(\tau)}}(\tau) &:= kn_{\delta(\tau)} \end{aligned}$$

The changes are specified as actions assigned to τ in the following way:

/p_delta_1=k_1, p_delta_2=k_2, etc.

Now that the implementation of the elementary building blocks of logical constructs has thus been elaborated, the implementation of the constructs as modelling and specification elements in their entirety is elaborated in the next section.

4.5.3 Implementation of the three distinctive logical constructs

λ_{sr} , λ_ℓ , and λ_p

Logical constructs λ_j are instantiated as Stateflow charts in Matlab Simulink, as shown in Figure 35a. A Stateflow chart is connected with other elements in a Simulink block diagram through data ports and event ports, and it has ‘internal’ data signals and events, which are not exchanged with other Simulink blocks. The external data and event ports, as well as internal data and event signals are specified and named using the Simulink model explorer (Figure 35b). Once signals have been entered there, their ports appear in the ‘Chart’ block.

There are four types of external event and data ports: input events, input data, output events, and output data. Whether a signal is ‘Input’ or ‘Output’ is specified in the Simulink model explorer under ‘Scope’. Input events are meter events $e_{met,i}$ and logistic events $e_{lgs,i}$. Only one port is available for all input events, which means that a multiplexer block (‘mux’ in Figure 35) must be inserted to merge multiple event signals. The orientation r_i of a meter event $e_{met,i}$ is specified in the Simulink model explorer under ‘Trigger’ as ‘Rising’ (↑), ‘Falling’ (↓) or ‘Either’ (↕). Logistic events do not have an orientation. This is specified under ‘Trigger’ as ‘Function call’. The input data ports are used for connecting the Stateflow chart to meter signals m_i and logistic values u_i , the output data ports for control signals p_i , duration data d_i , and logistic values u_j , and the output event ports for logistic events $e_{lgs,i}$.

For internal events and data the ‘Scope’ is specified as ‘Local’. As was discussed in 4.5.2, internal events are logistic events. Only explicit events have to be specified using the model explorer, since implicit events are generated automatically. Internal data are logistic values, which are used as intermediate results in the calculation of control values.

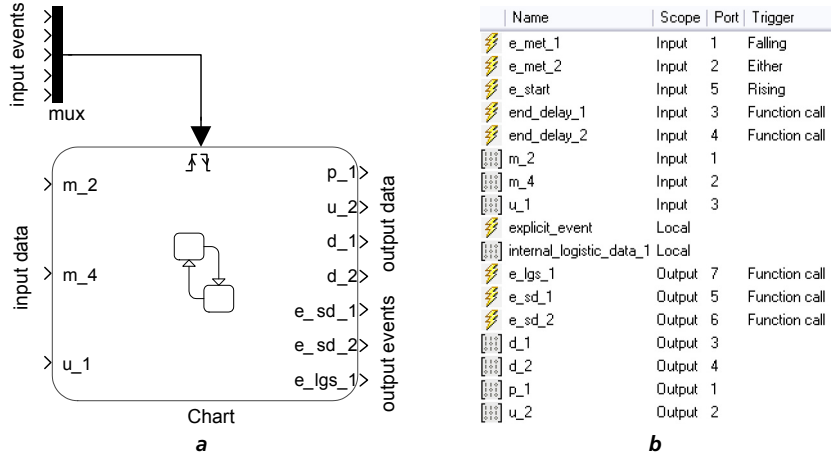


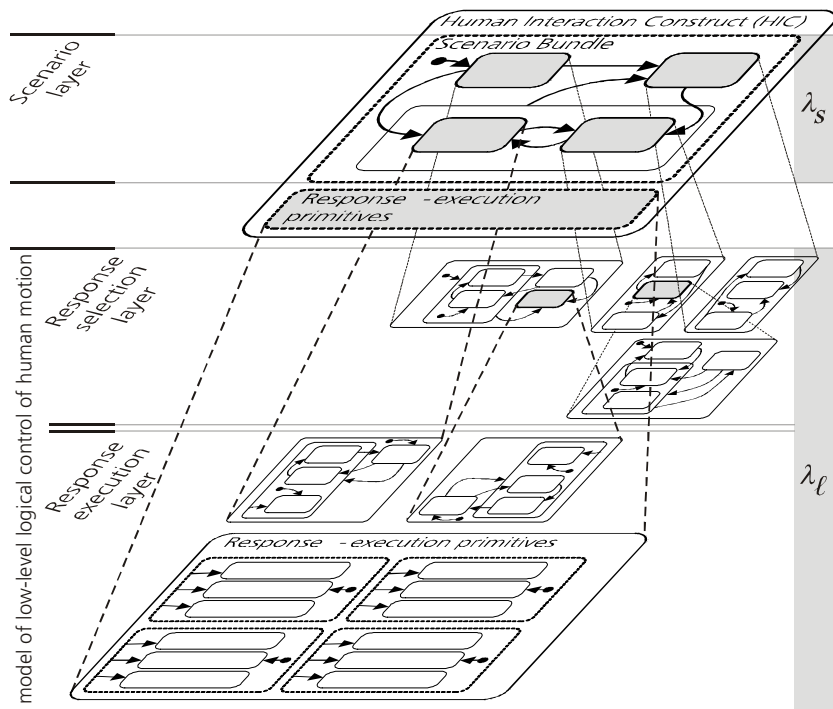
Figure 35. a: Example of a Stateflow 'Chart' block with external connections in Matlab Simulink. b: specification in the Simulink model explorer

There are three logical constructs: the scenario bundle λ_s , the model of low level logical human motor control λ_ℓ , and the procedure structure λ_p , the latter of which is optional since not all products have embedded software. These could be specified in the proof-of-concept implementation as three separate Stateflow charts as described above. However, since the constructs λ_s and λ_ℓ , which control human motions, are closely interlinked through logistic events sent from λ_s to λ_ℓ , the two have been combined into one human interaction construct, λ_h (see also 3.6.1) or HIC. This way, transitions in the two constructs could be synchronized by implicit events that do not need to be specified. Also, explicit events could be used throughout both constructs without having to specify them both as output of one construct and input of the other.

In the HIC, the model of low level logical human motor control has been further decomposed into two layers corresponding to the two levels of low-level motor control distinguished by Stelmach [172], i.e., response selection and response execution. Thus, by considering the scenario bundle as corresponding to the decision-making level above these two, the HIC can be considered as structured into three layers: the scenario layer, the response selection layer, and the response execution layer (Figure 36). The layering is realized using the subcharting functionality of Stateflow charts as explained in 4.5.1.

The instructions in the scenario layer specify the intentions of interactions only, and do neither refer to involved body parts nor to their motion patterns (e.g., 'reach for dial'). They represent decisions as transitions between states (e.g., a decision to change from state 'do nothing' to state 'pull lever').

The behaviour descriptions in the response selection layer address the involved body parts, the required/selected motions and how fast they are done (i.e., designer-specified general indications of velocities) without specifying how velocities, positions, etc. change during the motion (e.g., 'raise forearm fast'). The behaviours modelled in this layer appear as subcharts of states in the scenario bundle (e.g., 'retract upper arm' as one of the substates appearing in the subchart of 'pull lever'). Finally, to execute motion patterns, the behaviour descriptions in the response-execution layer specify quantitative changes in the control variables. These basic low-level control commands for the movement of each limb for each one of its degrees of freedom are represented by so-called *response-execution primitives* (Figure 36). These are basic low-level control commands for the movement of each limb in for each one of its degrees of freedom (e.g., move forward, rest, and move backward, with specified changes in angular velocity). Apart from response-execution primitives, the response execution layer may contain detailed models of specific motor routines, such as continuously adjusting orientations of body parts for balancing.



(Subcharting as in Figure 34)

Figure 36. Layered structuring of the human interaction construct (HIC)

4.6 Implementation of the signal conversion specification

In accordance with Figure 28 in 4.2.3, the proof-of-concept implementation of the signal conversion specification consists of the following signal conversion modules: (i) specifications for stimulus recognition, (ii) a the start event specification, and (iii) specifications for timing. In Simulink, the signal conversion specification is instantiated as a so-called subsystem, which appears as one block when viewed at a higher level. Figure 37 shows the contents of this block, which consists of further subsystems corresponding to the three types of modules mentioned above.

This example of a signal conversion specification contains an arbitrary quantity of meter values, meter events, delays, etc. The ovals to the left represent the input signals: (multiplexed) meter signals m_i , start-delay events $e_{sd,i}$, and duration data d_i . The ovals to the right represent the (multiplexed) output signals to the logical constructs: events $e_{start,i}$, $e_{met,i}$, $e_{ed,i}$ and condition values v_i , $\{v_i\} \supseteq \{m_i\}$.

The two *specifications for stimulus recognition* – one for the human and one for the artefact – convert meter signals to meter events. Figure 38 shows how specifications for stimulus recognition are instantiated. Meter signals $\mu_2(t)$, $\{\mu_i(t)\} \supseteq \{m_i(t)\}$ that have been selected as recognition signals are first processed by an event-generating function $f_i(\mu_1(t), \mu_2(t), \dots, \mu_n(t))$, which is depicted as a Simulink subsystem `Event_gener_func_i`, since it can be any purposeful operation

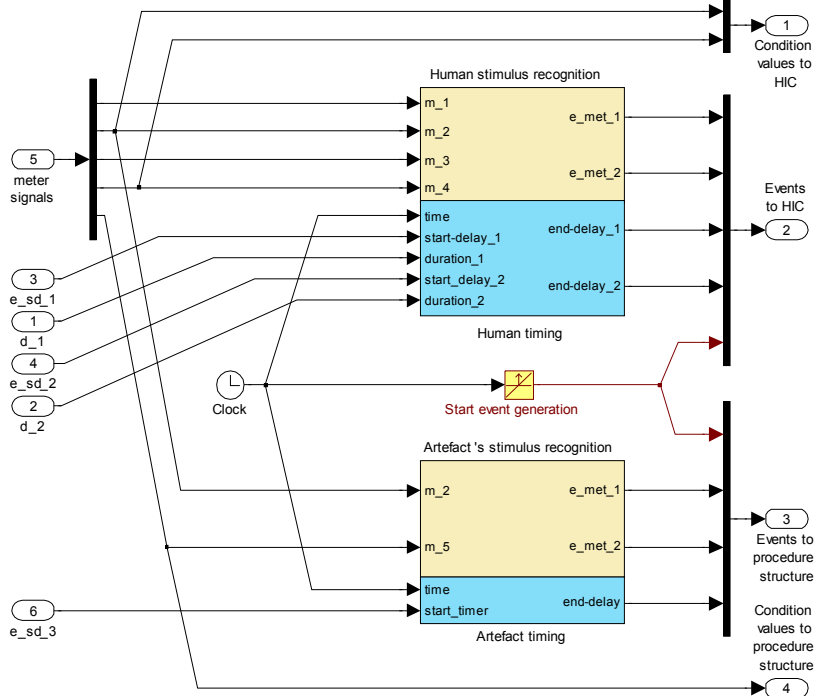


Figure 37. Example implementation of a signal conversion specification in Simulink

performed on one or more variables. Then each function result is led through a hit-crossing block that is specified by parameters for the orientation of the event, r_i , and its threshold value. When the threshold is crossed in the specified direction, the hit-crossing block generates the event according to the definitions in 3.6.2

The *specification of the start event* is instantiated in a similar way (but without an event-generating function), with a clock signal as its input. It is shown in the centre of Figure 37. The threshold value specified for the hit-crossing block corresponds to a delay in seconds between the user-generated start command for the simulation, and the actual start triggered by the start command⁵³.

The *specifications for timing* define the signal processing for the generation of delays in the control of motions in the human and the artefact, based on the time signal, a start-delay event $e_{sd,i}$, and a delay duration d_i . The specification is instantiated as follows (Figure 39). When the start-delay event is received, a time-out block (which is the equivalent of an event-generating function) starts counting back the remaining delay time, which is led through a hit-crossing block that generates $e_{ed} = e_{ed}(0, \downarrow, t_{remaining})$ when the remaining time hits zero. The time-out block uses a sample&hold block, which outputs the time signal, but 'freezes' its value when the start delay event is received. After the start-delay event, this output value plus the delay duration minus the actual simulation time

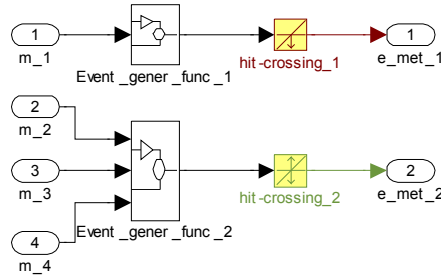


Figure 38. Contents of the human stimulus recognition block in Figure 37

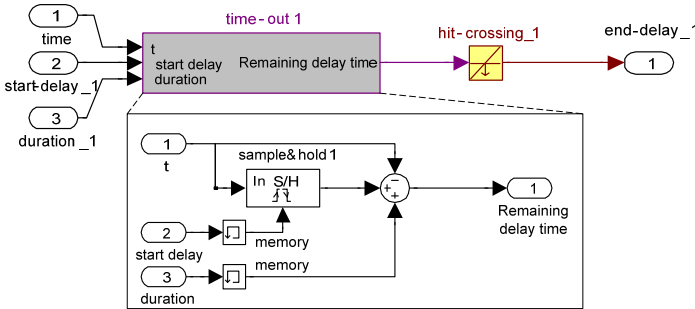


Figure 39. Specification of a delay

⁵³ This delay can be arbitrarily small, but it cannot be completely avoided. Time is a monotonically increasing function, and therefore, the orientation of the start event must be $r = \uparrow$. If the delay would be set to zero, no threshold is crossed because there is no simulation history before the start command.

equals the remaining delay time⁵⁴. The `memory` blocks⁵⁵ have been inserted to break algebraic loops that are not allowed in the control of physical simulations (see, for instance, [68]).

4.7 Interfacing the control and the simulation model

In the proof-of-concept implementation, the proxies that have been elaborated in 4.4-4.6 are brought together and connected in a Simulink block diagram. Figure 40 shows the block diagram in which the examples in Figures 33, 35, and 37 have been included, together with an example block representing a procedure structure. The examples have been devised so that all the modelling and specification elements defined in Chapter 3, and elaborated for the proxies in 4.4-4.6, are used, insofar they are ‘visible’ in block diagrams. This means that Figure 40 (page 106) can be considered a template for interfacing proxy constructs in general, considering the remark that the numbers of variables have been arbitrarily chosen, and that these are typically different for each case.

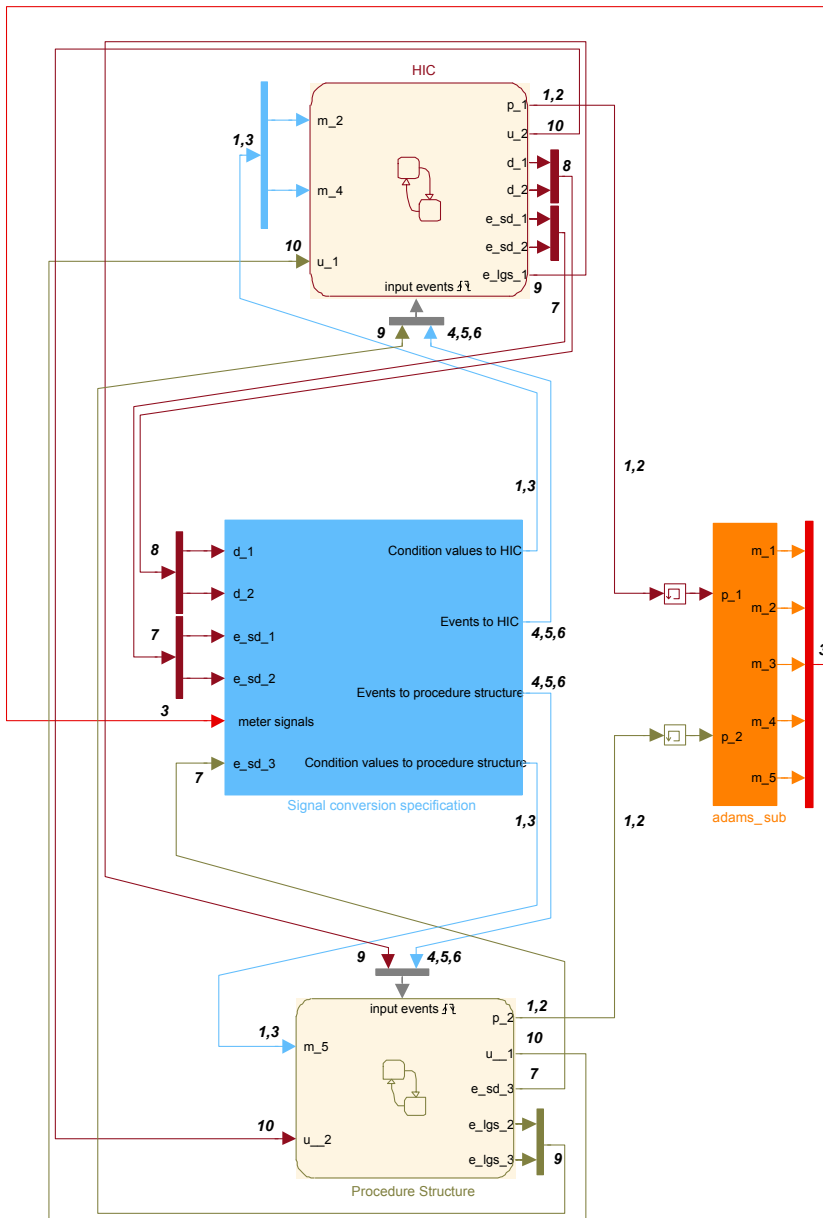
If the block diagrams and its (sub-)components have been completed, a simulation can be started by clicking on the ► button in any of the Simulink windows. The simulation runs until the total simulation time has elapsed, which is specified by the user in the Simulink block diagram window. There are various options for the visualization of simulations and simulation results. These are discussed and demonstrated in Chapter 5 by applying the approach to a sample product and simulating its use.

4.8 Overview of modelling elements, specification elements and their mutual relations in the proof-of-concept prototype

In Appendix 4, an entity-relationship diagram has been included to provide insight in how the various modelling and specification elements in the proof-of-concept prototype are related to each other.

⁵⁴ Actually, if the duration value d_i does not depend on processing in a logical construct, it is also possible to specify it by setting the threshold of the hit-crossing block to $h = -d_i$. This way, as it is the case for the artefact timing in Figure 37, it is not necessary to import a duration value from a logical construct.

⁵⁵ A Memory block outputs its input from the previous time step.



Signal numbering in italics refers to Table 6 on page 79.

Figure 40. Simulink block diagram in which all the constructs for resource-integrated simulation have been included and connected.

5 APPLICATION AND TESTING OF THE PROOF-OF- CONCEPT IMPLEMENTATION

5.1 Objectives

This chapter presents a series of sample cases that I elaborated for experimental application and testing of the proposed approach, based on the proxy implementation of the theoretical conceptual elements that have been elaborated in the previous chapter. The goal of testing has been to show that the developed theory is consistent and feasible, and to demonstrate its new functional affordances that are not available in current commercialized systems.

5.2 Approach

To test the proof-of-concept implementation with respect to the aspects described above, I applied the strategy of component-based application development. First, I developed four generic sample cases in which the approach was applied to elementary interactions: (i) dropping an object, (ii) throwing an object, (iii) reaching for an object at a given location, (iv) pushing and releasing a foot pedal, (v) pushing and releasing a button, and (vi) grasping, lifting, and carrying an object. Following the strategy of component-based application development, these components were defined to be reusable in other contexts and application cases. The six component cases resulted in basic scenarios with only a few transitions, which could be used to simulate the elementary interactions (e.g., Figure 41). To be able to test the applicability of the approach in the case of more sophisticated scenarios, three composite cases were developed in which elementary interaction components have been combined in a use process of a product.

In the first composite case, interactions (i) and (iv) have been combined to simulate the use of a pedal bin (Figure 42). In the second composite case, interactions (ii) and (iii) have been combined to simulate a human trying to throw an object into an open garbage bin (Figure 43), and reaching for the object if it lands outside the bin. By varying prescribed

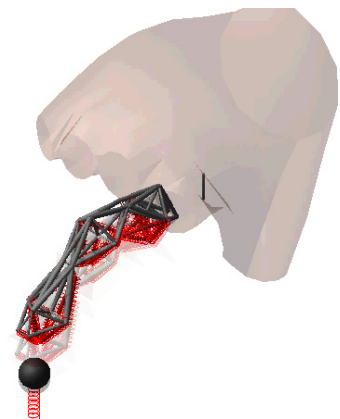


Figure 41. Scenario-based simulation of a basic interaction: pushing and releasing a button.

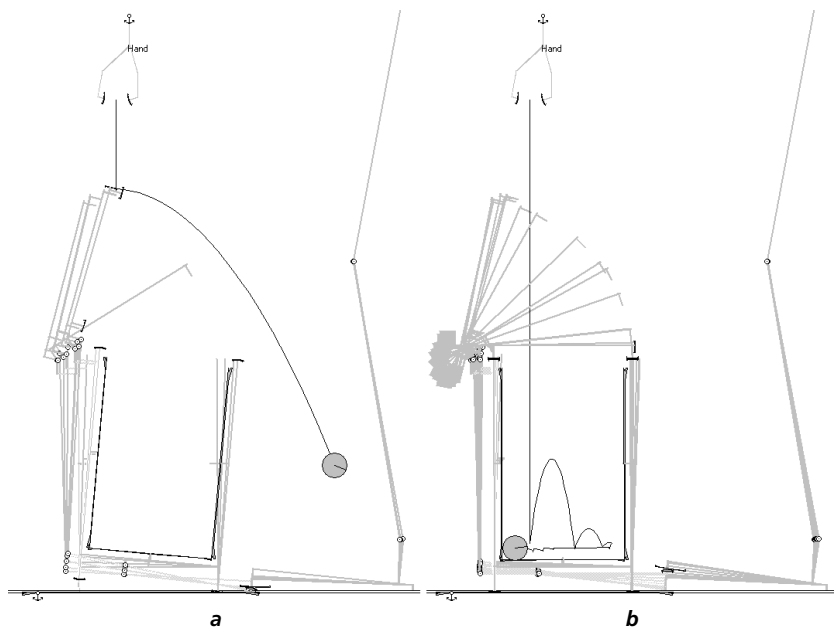


Figure 42. Simulations of using two versions *a* and *b* of a pedal bin

velocities in the scenario bundle, both paths in the scenario bundle (object lands inside or outside the bin) could be simulated. These first two composite cases were based on early, slightly different versions of the theoretical and implementation elements described in chapters 3 and 4. Also, a procedure structure to describe embedded logic in artefacts was not included. Detailed elaborations of the pedal-bin and open garbage-bin case can be found in [30] and [31], respectively.

In the third composite case, interactions (iii), (v), and (vi) were brought together to simulate a human customer retrieving a snack from a snack dispenser after having pushed a button. This composite case, and the component cases brought together in it, have been fully modelled and specified with the theoretical and implementation elements described in chapters 3 and 4. The elaboration in the remainder of this chapter concentrates on this second, more complex composite case, thereby including the elaboration of the three basic component cases on which it was based.

The conceptual design of the snack dispenser (Figure 44)⁵⁶ was created with the proof-of-concept implementation using Adams. In the regular use of this product the basic interactions that are involved appear as follows: (ii) reaching for the button, (iii) pushing and releasing the button, (ii) reaching for the snack, (iv) grasping the snack (which comes in a cylindrical package), lifting it, and carrying it. The dispenser has some built-in logic that controls the release of the snack to a customer, and which is described in a procedure structure. The snack

⁵⁶ The unlabeled parts in the figure are fixed parts for support and guidance of the snack

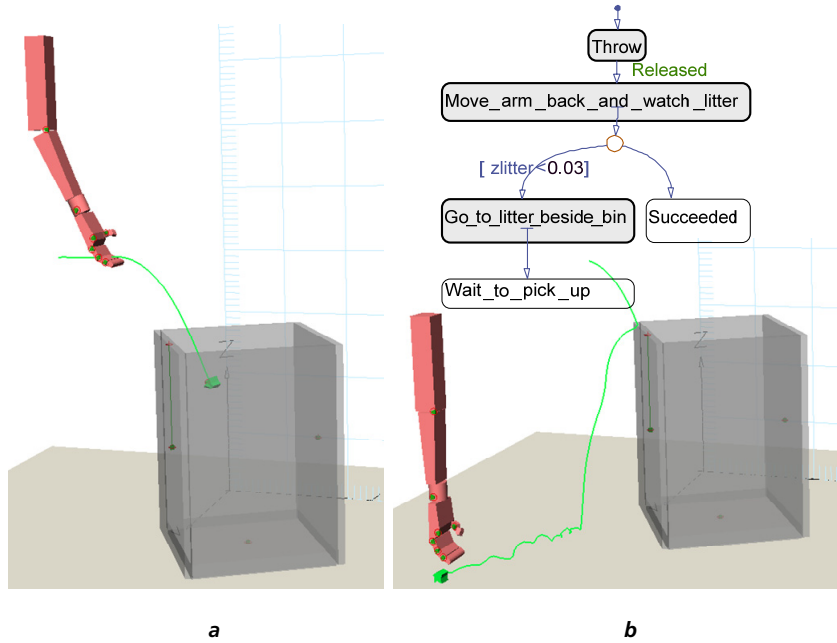


Figure 43. a. Simulation of throwing an object into an open garbage bin. b. End result of the same simulation after missing. The scenario bundle is shown at the top right.

must be kept cold, and therefore if the customer does not grab it within a given interval of time after pushing the button, the dispenser mechanism puts it back into storage. A retry loop that the customer performs if this happens is one of the alternative scenarios that I could include because of this.

5.3 Instantiation of behavioural simulation models

Since the investigation of the dispenser's operation and human interaction is made in conceptual design, I could use a simplified model architecture and detailing. This simplification in turn allowed reduction of the computation time and the effort needed for modelling.

For the snack dispenser and the snack I have only modelled those parts and geometric features that are directly involved in dispensing the snack (a simple cylindrical object), and interaction with the human. Furthermore, since only the human fingers have been considered as undergoing large deformations, all the components of the dispenser and the snack have been modelled as rigid by using simplified particle clouds as has been explained in 4.4.2.

For the human user, I have only modelled one arm and hand only and two fingers on this hand. From a geometric representation point of view, the fingers have been modelled as physically characterized particle systems to enable the large deformations needed for a firm grip on objects when grasping. To reduce simulation time, the resolution of particle clouds was kept low; i.e., only 81 par-

ticles were used for both fingers. Another simplifications applied to the human model was that no efforts have been put in obtaining or creating a completely realistic model that accurately describes the various physical properties and behaviours of the human body. A strategy of trial-and-error was applied, varying connection arrangements between particles, and varying coefficients of friction, damping, stiffness, etc., until a model was obtained of which the behaviour looked realistic.

For the whole model, two further simplifications were made. Firstly, the responsiveness of effectors has not been optimized to make their behaviours correspond to real muscles and actuators. To achieve that at least the response of the effectors was sufficiently fast and free of oscillations, the constants K_P and K_I (proportional and integral gain of PI controllers that convert prescribed velocities to forces, see 4.4.3) have been determined by trial-and-error and not been further optimized using tuning methods [e.g., 268]. Secondly, I have not considered assignment of entities to the levels component, assembly, and system. Although this is possible in Adams, as explained at the end of Section 4.4.2, it is mainly important for the organization of entities during modelling, but it would have had no effect on simulations or other outcomes of experimental testing.

5.4 Instantiation of control models and specifications

5.4.1 Specification and modelling of human control

Two logical specifications had to be defined to enable control of simulations of the snack dispenser, namely one HIC to control the interactions performed by the customer and one procedure structure to execute the control performed by the control mechanisms in the snack dispenser. These were specified and/or modelled using Simulink Stateflow R2007a. Since the focus of this thesis is on scenario bundle-based simulation, I will dedicate my attention mainly to the control of human interactions, which is the subject of this section. The procedure structure is presented in 5.4.2. As was explained in 4.5.3, the HIC has three layers: the scenario layer (which holds the scenario bundle), the response selection layer, and the response execution layer.

Figure 45 shows the scenario bundle, and Figure 46 shows the scenario layer together with

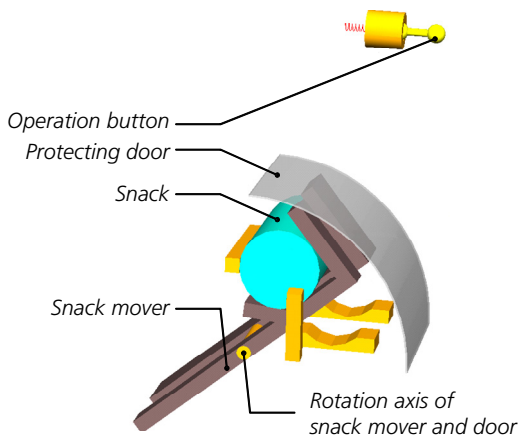


Figure 44. Concept design of a snack dispenser

the response-selection layer of the human interaction construct.

The scenario layer has been specified as follows. The ‘regular’ scenario describes a sequence of interactions, which commence from an initial stand-by state, and continue with activating the button, reaching for the snack, grasping it, and carrying it away. However, the designer may also specify ‘irregular’ scenarios. For instance, he can include human latency in the scenario bundle by inserting a state *hesitate*. It describes the situation when the customer hesitates for too long, and the snack dispenser has to put the snack back into storage space. Once the hand is close enough to the snack, a proximity sensor in the snack dispenser evokes an event within the procedure structure that prevents the snack from being put back. For this case (called *snack_gone*) the scenario bundle may also comprise a retry loop which includes another hesitation state (i.e. when

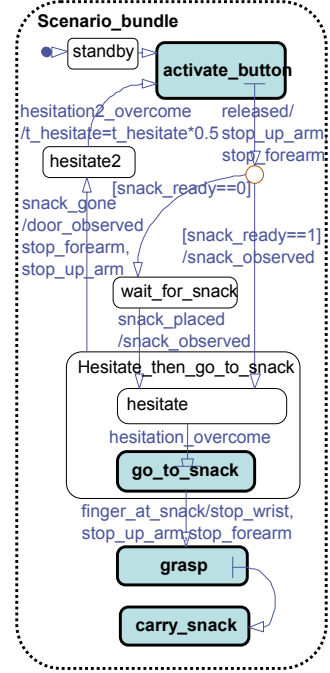


Figure 45. Scenario bundle of the dispenser

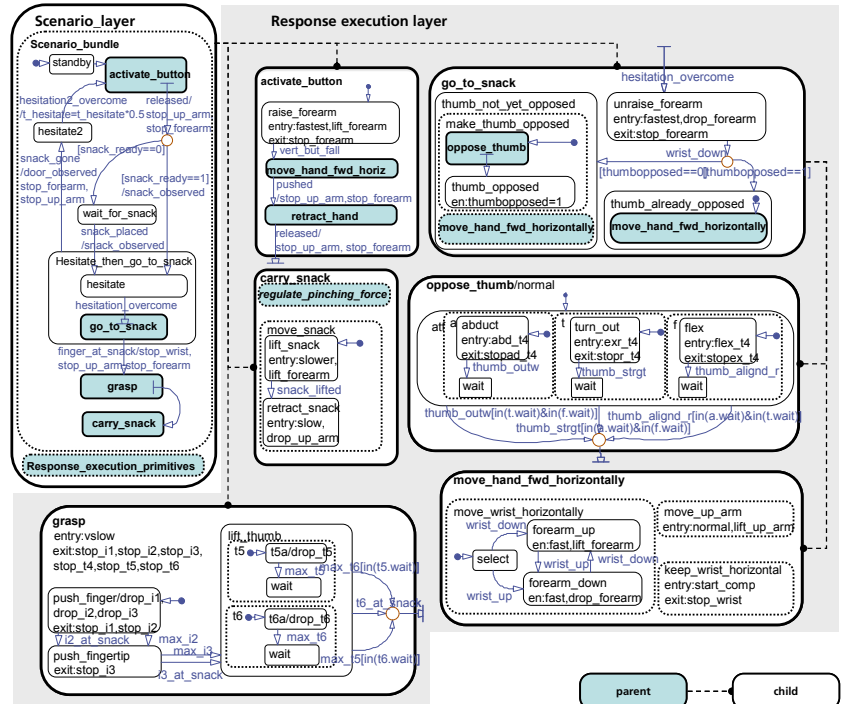


Figure 46. Scenario layer and response selection layer of human control in using the snack dispenser.

the customer is surprised to see the snack disappearing). The designer may also assume that, having learned from these happenings, the customer will react faster next time ($t_hesitate = t_hesitate * 0.5$).

Since, for the time being, achieving realistic human motion patterns was not my goal, I did not make use of the relevant knowledge of human motor science in specifying human response selection and response execution. At the same time, motion patterns are needed in the context of connecting the rather abstract commands embedded in the scenario bundle to the simulation algorithms. To resolve this contradiction, I adopted the policy of obtaining successful interaction results rather than absolutely realistic motions during the interaction. For instance, for moving the fingertip from A to B, I 'programmed' instructions that resulted in arrival at B, using motion patterns for which I did not care whether they were natural and efficient. In a future implementation, these workarounds are to be replaced by sophisticated instructions (or even continuous simulation algorithms) based on invariants and other findings from human motor studies. Since the current implementation is provisional, I just briefly summarize the most important issues of instantiating models for response-selection and execution in the elaboration of the sample case.

The response selection layer contains only those states that represent *results* of response selection processes, and not the procedures of selecting. Such procedures (see, for instance [231]) are needed to select interaction patterns that depend on variables (e.g., the size of the snack determines the grasping pattern). My reasoning has been that the selection procedures are not needed because these variables have already been defined in the simulation model, and are thus

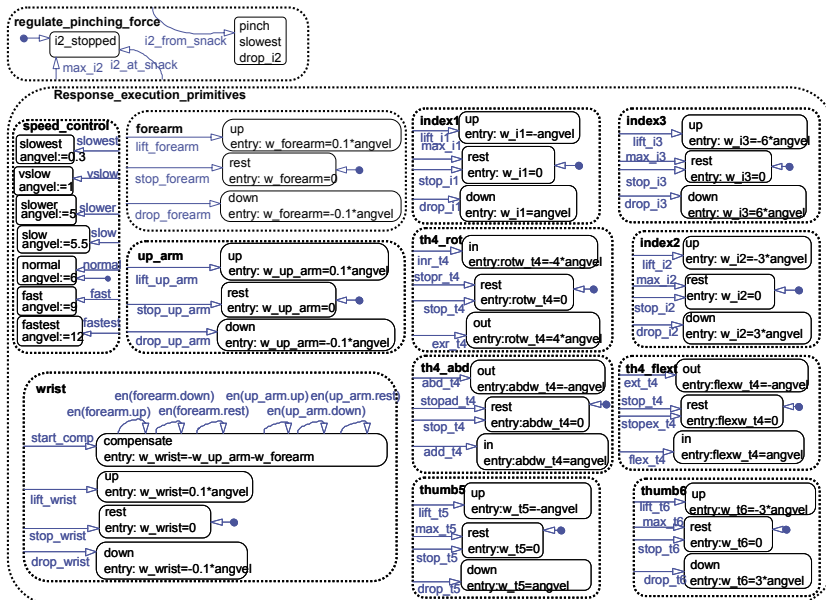


Figure 47. Response-execution layer of the HIC of the use of the snack dispenser.

known beforehand⁵⁷. Consequently, the response selection layer contains specifications of pre-selected responses, such as moving the hand forward horizontally, or preparing the hand for grasping, which includes opposing the thumb. In Figure 46, the states `move_hand_fwd_horiz` and `retract_hand` have not been decomposed because they are similar to `move_hand_fwd_horizontally`.

Figure 47 shows part of the specifications on the response execution layer, which include a routine `regulate_pinching_force` for clasping the snack, and the so-called response-execution primitives, i.e., basic low-level control commands for the movement of

each limb in one of its degrees of freedom. Firstly, there is a response execution primitive for speed control, which translates qualitative speed descriptions (slow, normal, fast, etc.) to quantitative ones for all states in which speeds have been specified. Secondly, there is a response execution primitive for each considered degree of freedom of each limb. in total. Each of the eleven degrees of freedom has three basic child states, two of them are for moving the limb up/down (or inward/outward), and one is for the limb's default rest state. Only for the wrist there is an extra state `compensate`, that maintains the orientation of the wrist while the arm is being bent. In this state, the angular velocity determining lifting/dropping of the wrist is compensated for the sum of the angular velocities of the upper arm and the forearm.

To (de)activate the motion of limbs, logistic events are included as actions in the scenario layer and the response selection layer. For instance, wherever an action contains the command `lift_forearm`, a transition to the child state `forearm.up` is triggered.

5.4.2 Programming of artefacts

A procedure structure has been defined as a logical specification that processes signals in the same way the embedded control mechanisms of an artefact would activate actuators based on input from sensors. Figure 48 shows the procedure

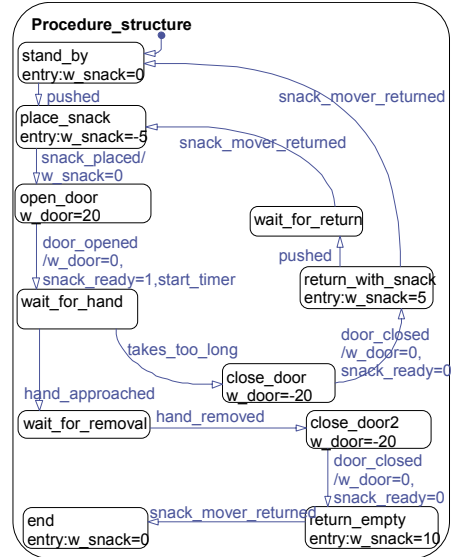


Figure 48. Procedure structure specifying the control mechanisms of the snack dispenser.

⁵⁷ This may be true in virtual prototyping of products, however, in other application areas of human interaction simulation, such as computer animations, virtual characters often have to deal with a variety of situations, in which case selection procedures cannot be left out.

structure of the snack dispenser. It controls, for instance, placing of the snack when the button has been pushed, and opening of the door after the snack has been placed. Some of the usual operations of snack dispensers were omitted to keep the procedure structure simple – in particular, the ability of delivering of multiple subsequent snacks and the collection of payments.

5.5 Instantiation of interfacing between control and simulation

Using Simulink, the logical constructs have been connected to the simulation model according to the implementation description in 4.7 (Figure 40). The signal conversion specification is a Simulink sub-construct according to the implementation description in 4.6 (Figure 37). The detailed Simulink models and specifications can be found in Appendix 5

5.6 Running controlled physical simulations

After finishing the models and specifications discussed in the preceding sections, I could run physical simulations by starting execution of the control instructions in Simulink. The commercial software packages, Simulink and Adams, generate user feedback that shows the changes in the human-artefact system during computation of the controlled simulation and afterwards. The feedback informs the user about (i) the progress through the logical constructs and (ii) the changes in the physical simulation model, both as a function of time. Simulink Stateflow shows the progress through the logical constructs by highlighting states and transitions when they are active. The user can follow this during the

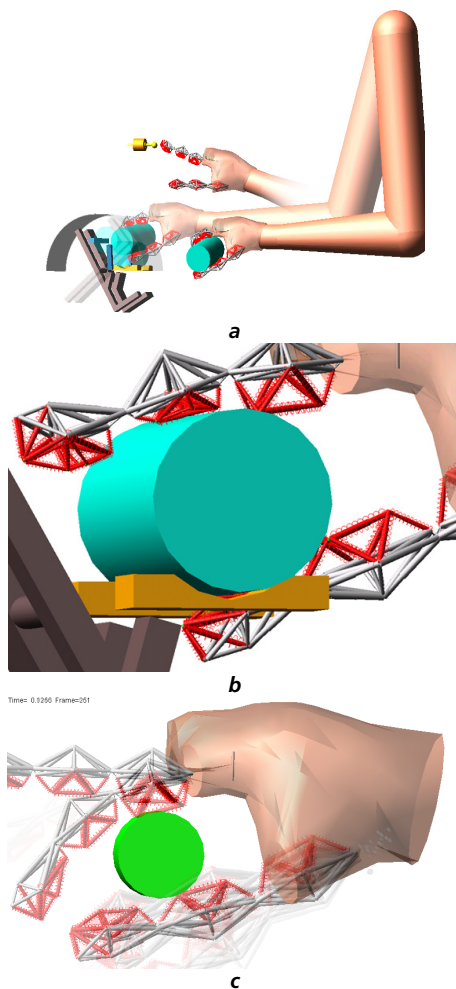


Figure 49. Compilation of simulation frames: a. overview; b and c. detailed views of grasping.

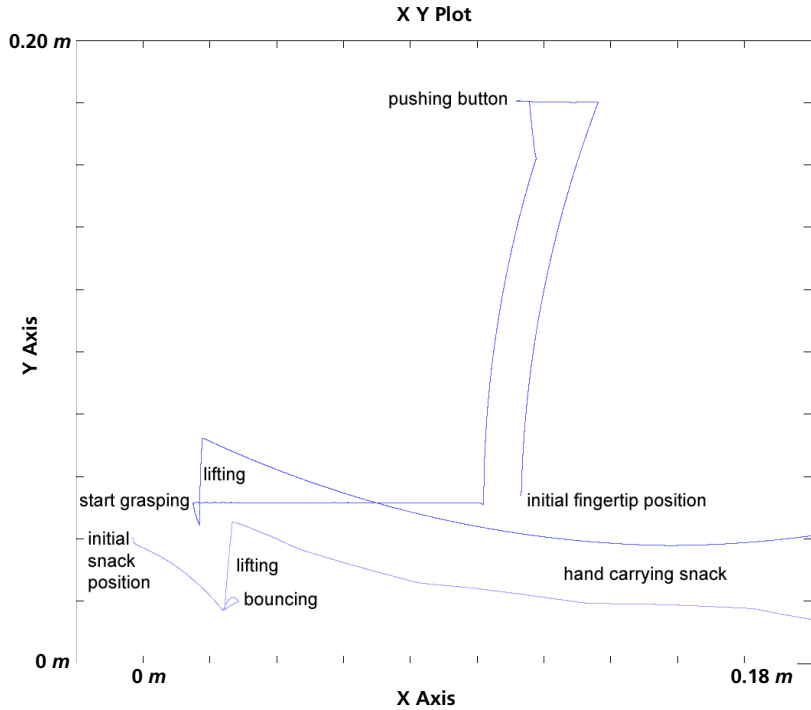


Figure 50. XY plots providing feedback during computation of the physical simulation.

computation but cannot revisit the course through the logical constructs afterwards.

Adams is also able to show the mechanical behaviours that it computes directly by animating the 3D view of the simulation model. However, this animation considerably reduces computational performance. Instead, I have let Adams perform its computations in 'batch mode'. This means that Adams is running its solver algorithm only, i.e., the computational algorithm that solves the differential equations of the dynamics behaviour. 3D animations of the interaction process were generated and visualized afterwards (Figure 49). To check the progress and status of interaction during the computation, *xy* plots of the position of two key points on the human body and the snack are produced by connecting Simulink *xy Graph* blocks to meter signals communicating the *x* and *y* positions of these points. In Figure 50, the top curve shows the positions of index-finger tip and the bottom curve shows the centre of the snack.

Disregarding the details of response selection and execution, I could simulate interconnected combinations of basic interactions through specifying them in the scenario structure: (i) reaching, (ii) pushing a button, (iii) releasing a button, (iv) grasping, and (v) carrying an object. Obviously, this is just a specific demonstrative set. For other application cases, different forms of basic interactions can also be defined after chunking the process. I use this set of interactions as a basis of evaluating the functional affordances of the scenario bundle-based specification

of a structure of physical simulation actions. As the demonstrative example shows, the proposed logical control mechanism made it possible to design various use-process scenarios and to monitor the conduct of these scenarios. In comparison with what is possible with conventional software⁵⁸, the control mechanism can be used to:

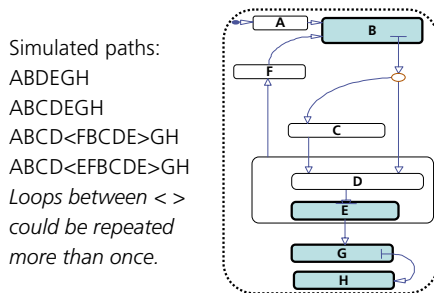


Figure 51. Simulated paths in the scenario bundle (cf. Figure 46)

- simulate multiple scenarios of concatenated interactions based on one scenario bundle.

Figure 51 shows four different paths that can be walked through within the specified scenario bundle. When the scenario bundle was processed, these paths were each individually operationalized. For instance, this allowed us to ‘play’ with the preset values of human latency (hesitation time), and with the preset values for the angular velocities that are imposed on the actuators in the snack dispenser. I have to mention that the four paths in Figure 51 correspond to three different paths in the procedure structure.

- control the physical simulation in the case of moderately varied artefact geometry and/or characteristics of the human by the same scenario bundle. My investigations explored that the proposed logical control mechanism can be applied to moderately varied artefact and human models, unless they required the introduction of new meter signals and control signals.

For instance, in my demonstrative application case, I confirmed this by varying the structural arrangement, in particular by changing the location of the button. When the position was raised or lowered by 25mm, the same scenario bundle could still be used to control the interaction simulation. The possibility of conducting simulation runs with different user characteristics by varying hesitation times was already mentioned above.

5.7 Discussion and conclusions of application testing

Considering the fast growing functional complexity of products, the snack dispenser is only a simple demonstrative application example. It combines however, the artefact and human models, the human interaction construct, and the procedure structure, and involves all actions the dispenser should perform.

As far as the benefits of application are concerned, I observed the following facts: (i) it allows digital exploration of the usability of a product and reduces the need for testing with human subjects; (ii) it offers a systematic approach, which facilitates the designer in considering human decision-making and its conse-

⁵⁸ The capabilities of conventional approaches are further elaborated in 7.2.

quences for interaction with products, and which connects to informal scenario-based approaches for mind-mapping the foreseen happenings; (iii) it allows testing moderate variations of artefacts (i.e., the product or artefacts in its use environment) without a need for additional changes in models and logical specifications; (iv) it allows tests with different characteristics of humans without a need for additional changes in models and logical specifications, and (v) it allows reuse of partial logical specifications for application in interaction simulation of similar products. Together, these benefits may help designers uncover imperfections in the design and anticipate uncommon ways of using the product.

On the other hand, I observed the following limitations and restrictions of application:

- Designers must learn to work with graphical representations for logical specifications before they can apply the approach.
- The physics simulation is susceptible to small changes in the thresholds of the events involved. The proxy model of low-level human motion control, which I had to tolerate in the absence of high-fidelity low-level human control models, seemed to be the most likely cause. Particular control instructions in this model involved small discrete motion corrections that could occur repeatedly at a near-constant frequency, thus coincidentally causing spring-damper elements to resonate. In some cases, this made computations result in extreme local forces and accelerations, causing unstable or crashing simulations. To get around this issue, the event-threshold values that I successfully used in the final simulations were fine-tuned by trial and error. However, in the case of an ultimate version of my simulation system, the designer is supposed to choose from established human control models offered by a predefined library, which does not require fine-tuning, and the stability of which has already been proven.
- Because of the abovementioned susceptibility, I was not yet able to test the application with variations of the human body model: each variation would have required additional fine-tuning, for which I lacked the time and resources.
- Animated feedback of the physics simulation during calculation is impracticable because of its negative impact on the simulation time. Even if this would be possible with the same performance as offered by the ‘batch’ computation that was used, the animation would still be so slow that it would be difficult to interpret directly. Although real-time performance is not needed, improvement in that direction appears to be necessary.
- The course of a simulated use process through the transitions and states of logical specifications cannot be stored for reviewing afterwards. It can only be followed directly, during the computation of the simulation.

Apart from the first mentioned item, it is believed that the main cause of these limitations and restrictions is that I used existing commercial software to develop a proof-of-concept implementation. These additional points can be addressed in future development of dedicated software.

6 JUSTIFICATION OF THE SUPPORTING THEORY

One of the aims of this PhD work has been to develop a proper theory that enables methodical investigation of use processes by simulation of successions of interactions, based on the principles of nucleus-based modelling and on scenario bundles. In this context not only the veracity and credibility of the theory are important, but also the implications of the theory in applications. With a view to these facts, logical justification of the elements of the proposed theory has been based on the concept of evidential reasoning with consequences. By this approach we can show where the theory can be accepted as empirically correct (proper), where it is partially correct, and where it fails.

Although this type of testing always goes together with the issues of comprehensiveness and induction, it can provide a reasonable scoping of the properness of the elements of the theory, which have been derived based on the assumptions formulated in the hypotheses. Below I have followed this approach. My intention has been *to scope the properness of the theory by reasoning about its consequences*, and thus to mark the boundaries of its veracity.

The consequences have been investigated by considering those conditions that incapacitate the theory from maintaining the assumed truth of the four fundamental hypotheses and from delivering the potential merits projected by these hypotheses. For each hypothesis the corresponding claims of the theory have been revisited. In terms of practical actions, I have identified those exceptions for which these claims are not true, in order to reveal the consequences of the theory in terms of its limitations. In my view this can lead to a qualitative demarcation of the field of problems for which the theory is proper and can be accepted. Implied by the hypotheses, the justification should target four topics: (i) connecting interactions with transitions, (ii) the logistics layer in resource-integrated models, (iii) scenario bundles, and (iv) the reasoning model of human-artefact interaction.

Connecting interactions with transitions. The first hypothesis states that connections between interactions can be specified as transitions in time, which conform to the concept of relations in nucleus-based modelling.

The proposed theory claims that all physical changes between interactions can be algorithmically described as changes computed from a given set of physically-based relations which is specified and instantiated by a nucleus (as postulated), and that all situational changes from one interaction to another (i.e., the connections) can be algorithmically described as transitions.

Neither the first hypothesis nor the theory is supported if, although two in-

interactions are connected in time, yet (i) the changes between them cannot be described as a transition specifying modifications in the set of physical relations, or (ii) there is a transition but it cannot be described because the sets of physical relations before and after the transition are unrelated to each other.

In the first case, if two interactions are connected in time and there is no transition between the two, then the second interaction is in fact a continuation of the first. In other words, this 'exception' describes one interaction in which physically-based changes can be computed from a given set of nucleus-based relations.

In the second case, if consecutive interactions are unrelated, then a radical random change should take place in the human-artefact system, e.g., the user is all of a sudden replaced by a different person, or objects (dis)appear suddenly, etc. While the description of such exceptions is possible for fictive processes such as, for instance, made up in animated movies, they never occur in real-life processes that designers want to simulate. On the other hand, we must not forget about two cases that might be exceptions to what is argued above: unrelated interactions occur when (i) the designer wants to skip certain interactions in the simulation of a use process because he deems them trivial or not interesting, or (ii) the designer intentionally makes changes to the human-artefact system *while* its simulation is running.

The logistics layer in resource-integrated models. According to the second hypothesis, the opportunity of specifying transitions manifests itself as a logistics layer on top of the behavioural simulation layer and the nucleus-based modelling layer. This results in resource-integrated models, the merit of which is that they permit prescribing, organizing, and connecting procedurally disjunct sequences of interactions.

The proposed theory claims that all transitions between interconnected interactions can be specified by a kind of logistic mechanisms, i.e., by a logical control mechanism, and that this logical control mechanism is able to communicate with the physically-based simulation by receiving and sending signals.

We have to see that the second hypothesis does not hold for use processes (i) in which transitions between interconnected interactions are not triggered by pure logistics, or (ii) in which signals between logistics and simulation cannot be defined/specified.

Regarding the first case, various use processes can be foreseen in which interactions are consecutive, but not connected based on logical process control. Typical 'uncontrolled' interventions in a use process occur as unexpected spontaneous events, such as power failure, sudden product failure, epileptic seizure of the user, stroke of lightning, etc. However, since we have already assumed that designers are able to conjure up user decisions that cannot be simulated based on behavioural models, then we can also assume that they are also able to conjure up 'uncontrolled' interventions and schedule them to happen at a given time. To take it one step further, these interventions in simulations can even be programmed to happen 'spontaneously' by specifying event-generating functions that contain stochastic components. Therefore it can be said that for these

exceptions, the theory still supports the second hypothesis.

Regarding the second group of exceptions that are not supported by the related part of the proposed theory, I could not identify use processes for which it is impossible to define or specify signals between the logical control of transitions and the algorithms of the physically-based simulation. Consequently, further investigations are needed to identify possible limitations implied by the claims in the respective part of the theory.

Scenario bundles. The third hypothesis says that human interactions can be treated as transition relations, which can in turn be specified in scenario bundles. The bundling allows designers to create organized sets of scenarios they have conjectured and want to test.

Concerning scenario bundles, the theory claims that all human decisions in use processes can be described and specified as state transitions.

If a use process involves human decisions that cannot be described/specified with state transitions, the respective part of the theory and the third hypothesis underpinning it do not hold. One can conceive an exceptional category of human decisions to which this applies. These are decisions that depend on storage and processing of past experience, typically in the use of products that require training or learning. Storage of experience does not comply to the assumption of momentary transitions in a use scenario. By considering and representing human decisions as state transitions, the theory does not provide clues how to specify scenarios for use processes in which the user gradually gains experience in operating a product. Describing this kind of learning-inclusive processes is even more difficult if it involves *implicit* or *tacit knowledge*.

Use processes in which learning explicit knowledge plays a role are for instance complex cognitive tasks in the use of computer software and programmable products such as video recorders. Usually the knowledge is explicitly available in a user manual, but a novice user will need some time before he or she can master all the needed forms of use.

Use processes in which learning implicit knowledge plays a role are, for instance, using skis, baseball bats and other sports equipment, complex driving tasks in the use of vehicles, or even kneading dough for bread using a kneading machine.

These aspects of human decision making and control are not, or not sufficiently, explained by the respective parts of the proposed theory.

Reasoning model of human-artefact interaction. To bring the aforementioned concepts together into a comprehensive use-process simulation approach, a simplified reasoning model of human-artefact interaction has been put forward in the fourth hypothesis. As a part of my theory, the assumed reasoning model arranges and connects the constructs used in scenario bundle-based simulation as well as those used in the functions performed by humans and artefacts.

Regarding the functions performed by humans, the theory claims that metabolism and other chemical processes do not have to be considered in human-artefact interaction. It also claims that the functionality of the perception organs can be surrogated by considering any computable change in the human-artefact

system as perceivable. Furthermore, the theory ignores that the human body might malfunction during the use process.

As a consequence, the corresponding part of the theory does not hold for use processes influenced by metabolism or other chemical processes, in which perception needs extensive consideration, or in which malfunctioning of the human body in the form of pathological phenomena needs consideration.

Regarding metabolism and other chemical processes, the reasoning model presumes that (i) the use process does not *directly* involve metabolic and other chemical processes, and does not depend on the effects of these: (ii) exhaustion of energy reserves, and (iii) changes in the properties of the human body. There are exceptional use processes for which presumption (i) does not hold - for instance use processes involving processing of human 'fuel' or waste, such as consumption of food products, the use of respirators, breathing equipment for divers, and astronaut suits. Products for which other chemical processes are important during use are, for instance, tanning beds and hair curlers. There are also exceptional use processes for which presumption (ii) is not true - for instance in use processes involving endurance: e.g., the use of military equipment, sports equipment and heavy-duty work equipment. Presumption (iii) does not hold for use processes that require consideration of growth of users (e.g., children's beds) or other changes in body properties (e.g., fitness equipment) – especially when simulation is used to monitor a virtual user during long-time use.

Regarding perception, the reasoning model presumes that all the information the human user needs from perception is readily available for evaluation conveyed by the scenario bundle. In real life, however, the assumption that all the knowledge is available is weak because in many cases it cannot be readily translated to logical conditions in a scenario bundle.

An example of knowledge that is typically unavailable is a visual stimulus appearing in the dark or behind the human's head. However, it seems possible to define 'filters' based on, for instance, a geometrically defined field of view in the physically-based human model.

What appears to be more problematic is dealing with perception of complex phenomena. These cannot be readily translated to logical conditions in a scenario bundle,. Here we have to think of phenomena such as:

- Vision input that requires advanced pattern-recognition skills (or computational processing to simulate such skills). For example, we can specify a perceived condition "door open" in the snack dispenser because it can easily be derived from values of variables in the mechanics simulation. However, we cannot specify a perceived condition "the curve ahead requires speed reduction" when using a car, because it needs evaluation of complex visual and motion patterns rather than that it depends on values of variables.
- Multi-channel input from various sense organs, requiring allocation of priorities: what is perceived and what is not? Situations where humans find themselves exposed to a sensual information overload (sometimes in combination with possibly imminent disasters) cannot be specified based on my simplified reasoning model. Examples of 'product use' where this applies are the con-

trol room of a nuclear plant and operating an aeroplane cockpit, where various alarms might go off at the same time.

Regarding malfunctioning or pathology, the reasoning model presumes that changes related to diseases can be ignored in the simulation of use processes. While a use process involving a user with a constant pathological condition can possibly be simulated by using an adapted user model, this is not true for a use process in which the pathological condition of the user changes in conjunction with the use process. In particular, this applies to various medical products, including equipment for physical therapy. As a conclusion it can be said that the current theory does not explain a set of phenomena related to human behaviour,

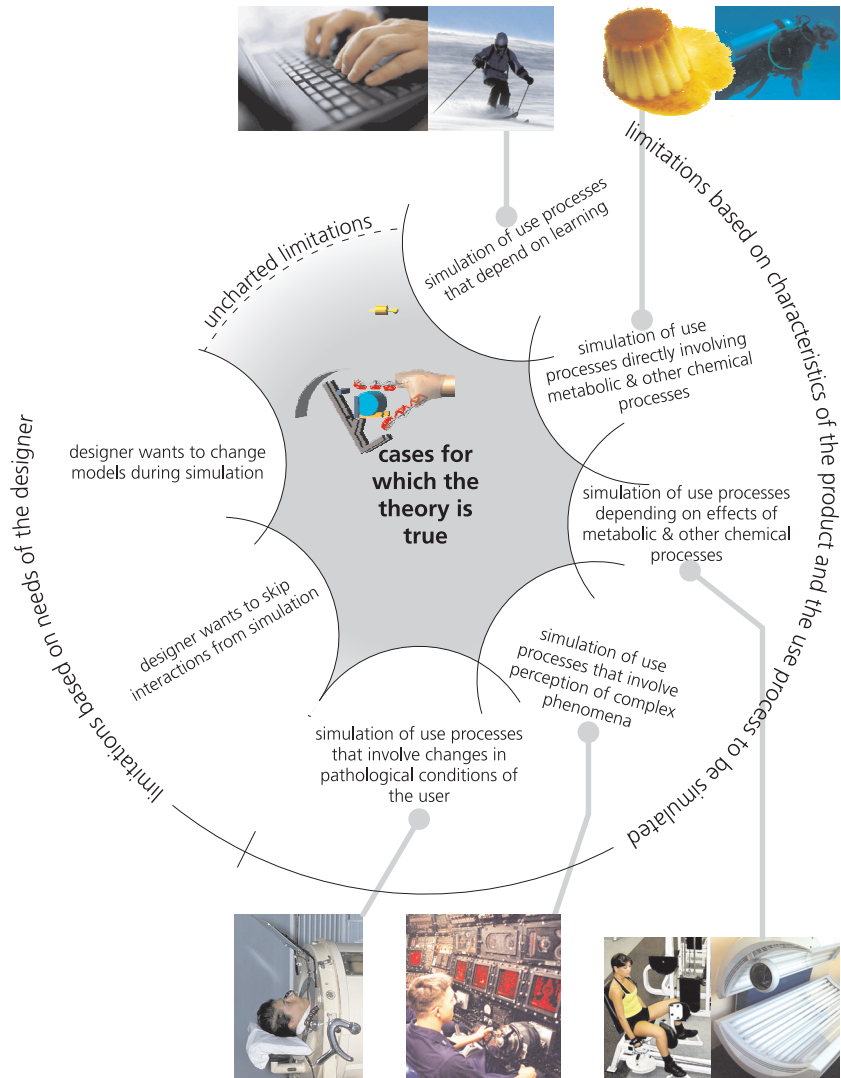


Figure 52. Identified limitations of the theory based on reasoning with its consequences.

and these phenomena can be considered as limitations of the current theory.

Conclusions of the justification. By reasoning with the consequences of the theory I was able to scope its properness and to identify the limits of its applicability. The result could convincingly be expressed by categorizing typical application cases for which the theory fails to support the assumptions of the hypotheses. My impression has been that reasoning with the implications of the theories allows various sorts of limitations, namely, (i) origination in the needs of the designer, i.e., possible needs of designers for which the theory does not offer a solution, and (ii) limitations based on characteristics of the product and the use process to be simulated, i.e., types of products and typical uses to which the theory does not apply. Figure 52 graphically depicts the identified areas that demarcate the valid application domains of the theory from the unsupported domains. As the figure shows, the inventory of limitations has not been exhaustive. To identify other, yet uncharted limitations, further systematic investigation is needed.

7 VALIDATION

7.1 Objectives

The underpinning strategy of validation of the proposed approach has been to show that the developed theory is consistent and feasible. This raised the need for implementation of a proof-of-concept system to the level of experimental testability. As presented in Chapter 4, programmable commercial tools have been selected and used as the basis of this prototype implementation. From the aspect of validation, my objective has been (i) to show that my approach lends itself to new functional affordances that are not available in current commercialized systems, (ii) to show the utility of the implemented system for product designs and to assess the convenience of use from technical aspects. A third objective would have been to encounter the benefits of application in relevant practical design processes. However, I had to face the fact that the proof-of-concept implementation allows only a qualitative estimation of possible benefits (which will be discussed in Chapter 8) rather than a thorough assessment.

The novel functional affordances offered by scenario bundles are discussed in 7.2. It is argued that a workaround can be constructed using conventional simulation approaches. This workaround can achieve similar functionality but lacks the flexibility that my approach may offer for efficient composition and execution of simulation runs.

Discussed in 7.3, the key technical aspects of convenience of use are (i) the time (and thus the effort) required from the designer and (ii) the time needed for a computer system to perform scenario bundle-based simulations. In addition to the functional comparison it is shown that even quantitative indices can be derived. Based on these indices it can be pointed out that the new approach offers advantages in terms of required effort and time. Subchapter 7.4 wraps up this chapter with concluding remarks.

7.2 Extended functional affordances of scenario-bundle based control of simulations

The concept of scenario bundle-based simulation control has facilitated the realization of functions that are not yet readily available in conventional simulation software. According to my literature survey, the systems currently used in virtual prototyping usually concentrate on conducting physical simulation. Some systems also include scheduling actions, but the combination of both has not been applied so far in the field of product design. I proposed and implemented the concept of scenario bundles, which can be used to interconnect simulations and hence to investigate alternative interactions. In general, scenario bundles make it possible to define constructs for interaction simulation in a context-sensitive way.

On the other hand, they are not sensitive to minor modifications of the elements of the physical simulation model, such as the model of the human, the product, or the environment. Finally, they introduce a kind of modularity, enable object-oriented thinking in designing similar processes, and facilitate the reuse of simulation resources (codes, algorithms, and simulation resources).

In order to provide evidences, I have compared the proof-of-concept implementation with other, conventional simulation software mentioned in 4.3.3., namely, with Virtual.Lab, SIMPACK, and CAMELview. According to the comparison, all these competing systems could be used instead of Adams in my proof-of-concept implementation because they allow external control through linking with Simulink. However, a *conventional* approach that I can use to benchmark my approach against should be aimed at simulation of connected interactions based on additional modelling elements *within a model* that has been instantiated with the conventional simulation system. To deserve the label *conventional*, the approach should *not* be based on specifying of logical constructs according to the definitions and specifications in Chapter 3 and 4, as it might be possible using the built-in logical expressions⁵⁹ that most of the systems offer. Specifying logical constructs in that manner would merely be an alternative implementation approach for scenario bundle-based simulation. As an implementation it would only be different in that it employs a less user-friendly⁶⁰ representation as a substitute of the graphical notations employed by statecharts, Petri nets, etc. Moreover, since their evaluation frequently causes mathematical singularities that lead to failing simulation computations, the use of built-in logical expressions in conventional multibody simulation systems is not recommended [269]⁶¹.

The traditional way to connect the individual interactions is to define intermittent constraints [259]. This conventional approach is based on consideration of the points in time at which values must be changed using Heaviside step functions, which is possible with all the investigated systems⁶². In terms of computation, it means that changes in meter variables are not evaluated, and no logic is applied. In this case, the problem is that transition times are not known beforehand. It means that all transition times have to be found systematically by running repeated simulations. In other words, in order to find the time $t(n)$ of the n th transition, a simulation must be run in which all the preceding transitions $t(0), \dots, t(n-1)$ have already been specified. To define consecutive transitions for the same control parameter at multiple points in time, the step functions describing the relative changes must be superimposed by linear addition. The result is a

⁵⁹ e.g., for a transition triggered by an event with orientation \uparrow and threshold h_i : $IF [m_i > h_i] THEN [p_i := new_p_i]$, where m_i is a meter value and p_i is a control value. Adams, Virtual.Lab, and CAMELview allow such expressions in the functions that specify prescribed forces and torques.

⁶⁰ for references supporting this statement, see 2.5

⁶¹ See also: Adams/View online user manual, "best practices"

⁶² Typically, to avoid mathematical singularities, variations on step functions are used that 'soften' sudden changes in signals

sequential control instruction for *one scenario*, which is *specific for one variation* of the artefact/user physical model. Any change in these models may result in different transition times. Consequently, if one wants to investigate any other course of the simulated use process, the whole procedure must be repeated. Therefore, my conclusion has been that the conventional approach, which can be realized in any of the investigated commercial multibody simulation systems, offers poor support for composing and executing interconnected simulation runs. On the other hand, we can efficiently handle multiple scenarios and multiple variations of the artefact/user model with scenario bundles.

7.3 Convenience of use from technical aspects

Though convenience of use of a design software system is a holistic concept, it can be expressed (quantified) in terms of how much effort and time are required from designers to apply it. This can be compared by benchmarking with conventional methods and tools. Benchmarking of the preparation effort and the simulation time was based on comparing (a) a proof-of-concept implementation based on proxy software package *X* as a proxy for the logistics layer and software package *Y* as a proxy for the behavioural layer with (b) the conventional approach with intermittent constraints as described in 7.2, using software package *Y* only. My reasoning has been that regardless the choice for software package *Y*, which is used on both sides of the comparison, this setup is sufficient to reveal the relative advantages and disadvantages of introducing control over simulations by using scenario bundles. By making assumptions about correspondences between various preparation activities and about computational complexity of simulation models and logical constructs, I was able to reason about benchmarks independent from the choices for *X* and *Y*. Since some quantitative experimental data was available about simulation computations, an additional concrete comparison could be included for the simulation time, in which $X = \text{Simulink} + \text{Simulink Stateflow}$ and $Y = \text{Adams}$.

In my evaluation of convenience, I have distinguished (i) a preparation efforts indicator and (ii) a simulation time indicator. Another aspect of convenience is the cognitive load raised on designers at using the system to specify control over simulation processes. This assessment however does not make sense when the fully fledged interface is not yet available.

Preparation efforts indicator (PEI). We can start from the fact that (a) an initial simulation model is above all necessary, and that we also need to specify (b) those variables that need to be controlled. The time spent on these activities has not been considered in benchmarking the time related to preparation because they appear on both sides of the comparison between the scenario bundle-based approach and the 'conventional' approach. In the case of the conventional approach with intermittent constraints (7.2, page 126) the further preparation activities are as follows: (i) define a Heaviside step function for each transition and (ii) superimpose the step functions on those variables that undergo multiple transitions. In the case of my approach, the additional activities are (i) crea-

tion of logical specifications, (ii) specification of meter values and (iii) connecting the logical control specification to the simulation model. The most obvious measure to express preparation effort is the number of person-hours elapsed. However, this measure is not the most appropriate if the proof-of-concept implementation lacks a dedicated user interface for preparation of logical control and physics simulation. Therefore, I introduced the preparation efforts indicator (PEI) that expresses preparation efforts in terms of the total number of required activities (N_A):

$$\text{PEI} = \frac{N_A(c)}{N_A(s)}$$

where c refers to the conventional approach and s refers to the scenario bundles-based approach.

In both cases, the total number of required activities (N_A) depends on the total number of defined transitions, N_T . It can clearly be seen that for the conventional approach:

$$N_A(c) = N_T(c), \quad (1)$$

if we neglect the effort needed to superimpose step functions. In this case, the total number of transitions $N_T(c)$ is the number of transitions actually occurring during simulation of an investigated scenario. Now let us assume that we want to apply the same scenario to n variations of the simulation model, and let us assume that the scenario is indeed executable for all these variations. Furthermore, it is assumed that to change a variation into the next investigated variation, one additional preparation step is needed (i.e., the designer is applying incremental modifications). Thus, the preparation efforts indicator can be reformulated for investigation of n variations of the simulation model as:

$$\text{PEI} = \frac{N_A(c_n)}{N_A(s_n)} \quad (2)$$

where c_n refers to the conventional approach and s_n refers to the scenario bundles-based approach.

Since all the transitions need to be redefined for each model variation, the number of preparation activities for the conventional approach is

$$N_A(c_n) = N_T(c) + (n-1)(N_T(c) \pm 1) = n \cdot N_T(c) + n - 1 \quad (3)$$

This equation takes into consideration that the extra preparation step to change the simulation model is not needed for the initial model.

In my approach, N_A depends not only on the number of transitions, N_T , in the logical specifications and on the number of investigated model variations, but also on (i) the number of states, N_S , and (ii) the number of meter variables, N_M , to be defined, and (iii) the number of connections, N_C , to be established between the control specification and the simulation model. On the other hand, with the exception of creating each new variation of the simulation model, all these activities need to be carried out only once:

$$N_A(s_n) = N_T(s) + N_S(s) + N_M(s) + N_C(s) + n - 1 \quad (4)$$

To be able to compare $N_A(c_n)$ and $N_A(s_n)$, correspondence between the terms of which they are composed has to be established. One can start out from the fact that in both cases a number of transitions is specified. This number of transitions is not necessarily the same for both cases. In the case of the conventional approach, the arrangement of transitions in time is always serial, which corresponds to a single scenario. In the case of the scenario-base approach, however, the designer can specify an arbitrarily large number of alternative paths by the scenario bundle. These in turn can include an arbitrarily large number of transitions, states, etc. At first sight, this would increase the number of preparation actions needed, i.e., $N_T(s) > N_T(c)$. However, to achieve a fair comparison it is assumed that the designer starts creating logical specifications with one serially arranged path only⁶³, and that based on investigation of obtained simulation results alternative paths may be added afterwards. This procedure applies to the conventional approach as well, therefore the consideration of adding alternative paths has not been considered in the comparison. Therefore, I will assume that $N_T(s) = N_T(c) = N_T$. As a conservative estimate, we can typically assume that:

- $N_S(s) \approx N_T$, since a transition always connects two states. The observation that a state can have more than one ingoing and one outgoing transition does not apply if we consider only one path for each logical specification;
- $N_M(s) \leq N_T$, since one meter variable can trigger more than one transition, but it is unlikely that a transition is triggered by more than one meter variable at the same time – assuming that meter variables that have been composed based on multiple simulation variables count as one;
- $N_C(s) \leq 2 \cdot N_T$, assuming that the number of connections approximately equals $N_M(s) + N_E(s) + N_D(s)$, where $N_E(s)$ is the number of events and $N_D(s)$ is the number of data values, where $N_E(s) + N_D(s) \approx N_M(s)$, so that $N_A(s) \leq 5N_T$.

Now we can include the above expressions in (4) and then in (2), which gives:

$$PEI \geq \frac{nN_T + n - 1}{5N_T + n - 1} \quad (5)$$

The graph in Figure 53 shows the minimum value of PEI, $PEI(min)$, as a function of n for different values of N_T . Since (5) is an inequality, the possible values of PEI are on or above the lines in Figure 53. The scenario bundle-based approach requires less preparation effort in those cases where five or more model variations are investigated. The scenario-bundle based approach is also advantageous with respect to the number of transitions in the scenario, but only up to a certain limit. For large values of N_T , the minimum value PEI approaches $PEI(min) = n / 5$.

⁶³ Implicitly, multiple concurrent serially arranged paths are also allowed under this assumption. In the conventional approach, these would be specified as one serial arrangement of transitions. For example, let us consider two concurrent serial paths A and B in a scenario bundle, with transitions A1, A2, A3, and B1, B2, respectively. In a simulation these might for instance occur in the order B1-A1-A2-B2-A3-B2, which is also the linear arrangement in which the transitions should be specified with the conventional approach.

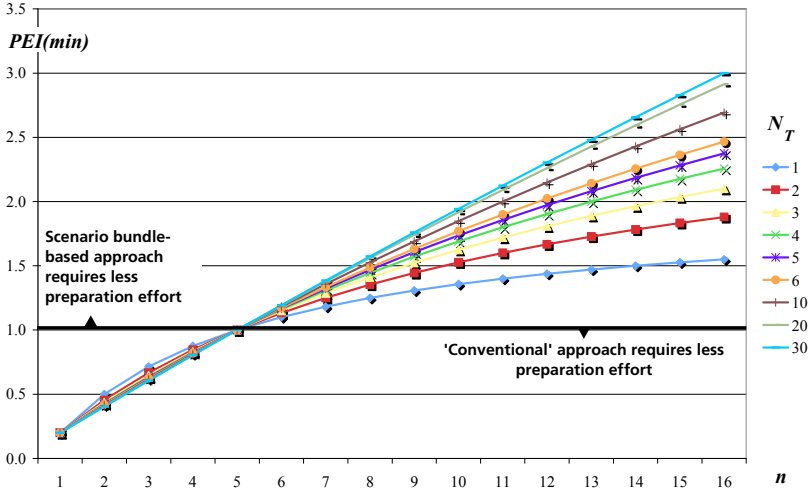


Figure 53. Minimum value for the preparation efforts indicator, PEI, as a function of the number of model variations, n , and the number of investigated transitions, N_T .

However, it has to be noted that the conventional approach needs additional time for the repeated simulation runs for each individual variation, which increases with the number of transitions. This is regarded as simulation time rather than as preparation activities and considered in the simulation time indicator, which is discussed next.

Simulation time indicator (STI). An accurate proportional measure for the simulation time is the total CPU time, T . By splitting it up into $T = T_{sim} + T_{ctrl}$, where T_{sim} is the CPU time for simulation and T_{ctrl} is the CPU time for control, a comparison with the conventional approach can be made. In the case of the latter, repeated simulation runs are required. I have defined the STI as:

$$STI = \frac{T(c)}{T(s)} = \frac{T_{sim}(c)}{T_{sim}(s) + T_{ctrl}(s)} \quad (6)$$

where c refers to the conventional approach (in which no computations are allotted to control), and s refers to the scenario bundle-based approach.

Let t_i be the time needed to simulate the processes between transitions τ_i and τ_{i+1} , then

$$T_{sim,total} = \sum_{i=1}^{N_T+1} t_i$$

is the total time needed to simulate the whole use process from the first to the last, (N_T) th, transition, plus the state after the last transition, in one run.

Let $\overline{t_{sim}} = \frac{T_{sim,total}}{N_T + 1}$ be the average simulation time between two consecutive

transitions, so that

$$T_{sim}(s) = (N_T + 1) \overline{t_{sim}} = T_{sim,total} \quad (7)$$

$$\text{and } T_{sim}(c) = \sum_{i=1}^{N_T+1} i \cdot t_{sim} = t_{sim} \cdot \sum_{i=1}^{N_T+1} i = \frac{T_{sim}(s)}{N_T+1} \sum_{i=1}^{N_T+1} i = \frac{1}{2}(N_T+2) \cdot T_{sim}(s) \quad (8)$$

Then, if $T_s(s)$, $T_c(s)$, and N_T are measured from an actual simulation performed by the proof-of-concept implementation, the STI for that particular simulated use process can be calculated from (6)-(8) (Note that to calculate the STI for a particular sample case, simulation runs based on the conventional approach are not needed):

$$STI = \frac{\frac{1}{2}(N_T+2) \cdot T_{sim}(s)}{T_{sim}(s) + T_{ctrl}(s)} \quad (9)$$

which can be rewritten as

$$STI = \frac{N_T+2}{2+2\zeta}, \quad \zeta = \frac{T_{ctrl}(s)}{T_{sim}(s)} \quad (10), (11)$$

The ratio ζ is specific for each application case, since T_{ctrl} depends on the computational complexity of the logical constructs, i.e., on the evaluation of transitions, and T_{sim} depends on the computational complexity of the simulation models, i.e., on the evaluation of motion equations for entities in the simulation model. By expressing ζ as a function of the number of movable parts in the simulation model and the number of transitions, it can be shown that if the number of parts is larger than a particular minimum, the total simulation time for scenario bundle-based simulation is always shorter. To find a suitable expression for ζ , let us assume as a conservative estimate that

- (i) there are n_{mov} entities in the simulation model of which the degrees of freedom have not been completely constrained so that they can move;
- (ii) for each of these n_{mov} entities at least one algebraic expression needs to be evaluated during simulation (*Regardless of the simulation algorithms used – Adams, nucleus-based or anything else– this is a very pessimistic estimate. It is based on the assumption that all the moving entities have only one degree of freedom, and that the simulation algorithms can effectively exclude all non-moving entities from computations*);
- (iii) the computational evaluation of (a) one time increment for an algebraic expression in simulation and (b) the logical processing of one transition take the same amount of CPU time, t_{eval} (*in reality, evaluation of motion equations is more time-consuming*);
- (iv) the time increment for simulation computation Δt_{sim} and the communication interval between simulation and control Δt_{ctrl} are the same, Δt_{sim} (*in reality, typically, $\Delta t_{sim} < \Delta t_{ctrl}$*);
- (v) after each communication interval Δt_{ctrl} , one transition takes place, which in combination with the previous assumption suggests that algebraic expressions are equally often evaluated as are transitions (*in reality they are evaluated more often*),

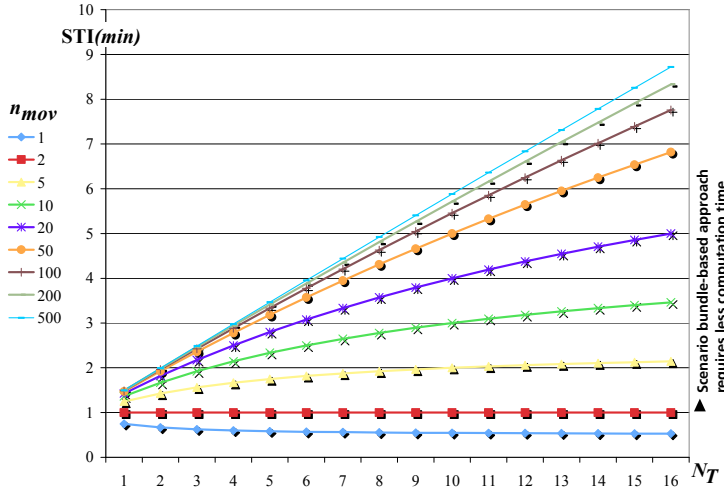


Figure 54. Minimum values for the simulation time index, STI, as a function of the number of moving entities in the model, n_{mov} and the number of investigated transitions, N_T .

$$\text{then } \zeta = \frac{T_{ctrl}(s)}{T_{sim}(s)} \leq \frac{N_T \cdot t_{eval}}{n_{mov} \cdot t_{eval}} = \frac{N_T}{n_{mov}} \quad (12)$$

If this expression is substituted in (10), then for $n_{mov} = 2$, the minimum value $STI(min) = 1$, and generally for $n_{mov} > 2$ it is true that $STI(min) > 1$. Only for $n_{mov} = 1$, the fact that it does not require logical evaluation of transitions puts conventional approach at an advantage. For large values of n_{mov} , the minimum value of STI approaches $STI(min) = \frac{1}{2}N_T + 1$, which implies a linear increase with the number of investigated transitions. The graph in Figure 54 shows the minimum values of STI, $STI(min)$, as a function of N_T for different values of n_{mov} . Since (12) is an inequality, the possible values of STI are on or above the lines in the figure.

To obtain an impression of the credibility of the above findings, I calculated the STI from my test results with the compound sample case, in which the use of a snack dispenser was simulated. I investigated the simulation corresponding to the path ABCDEGH in Figure 51. According to the performed simulation, the duration of such use is 2.9s. Simulating it required 48:02 minutes of CPU time⁶⁴ on a PC with AMD Athlon 64 X2 2.81 GHz dual core CPU running Microsoft Windows XP SP2 (using two threads for the Adams 2007 simulation). During the controlled simulation, Adams needed $1.40 \cdot 10^3$ s of CPU time (48.6% of the 48:02 minutes total CPU time), while according to the Windows Task manager Simulink used approximately 4.2% of total CPU time. This means that $T_{sim}(s) = 1.40 \cdot 10^3$ s and $T_{ctrl}(s) \approx 121$ s. The number of transitions that took place, N_T , could be counted

⁶⁴ including background processes not related to the simulation

for the scenario layer and the response selection layer: $N_T(\text{scenario}) + N_T(\text{response_sel}) = 21$. The number of detail-level transitions that took place in the response execution layer involved many small repetitive corrections that have not been taken into consideration. Substituting the above numbers in equations (6) and (8) gives:

$$\text{STI} = \frac{T_{sim}(c)}{T_{sim}(s) + T_{ctrl}(s)} = \frac{T_{sim}(s) \frac{N_T + 2}{2}}{T_{sim}(s) + T_{ctrl}(s)} = \frac{1.40 \cdot 10^3 \cdot 11.5}{1.40 \cdot 10^3 + 121} = 10.6$$

which means that, considering that not all transitions were taken into account, the 'conventional' approach with its repetitive simulation runs would have taken a total amount of CPU time that is larger by at least a factor 10.

The number of moving entities⁶⁵ in the simulation model is $n_{mov} = 98$. Checking for the value of the ratio ζ according to (12) appears to confirm that

$$\zeta = \frac{T_{ctrl}(s)}{T_{sim}(s)} = \frac{122}{1.4 \cdot 10^3} = \mathbf{0.087} \leq \frac{N_T}{n_{mov}} = \frac{21}{98} = \mathbf{0.21}$$

even though not all transitions have been taken into account.

In addition to the performance comparison with conventional simulations, I have also considered comparing the simulation time with the duration of the real-time process. On average, in the case of simulating the abovementioned path ABCDEGH with the snack dispenser, the simulation needs more than 16 minutes (48:02min / 2.9s) to compute each second of simulated behaviour. Based on a ratio $T_{ctrl}(s)/T_{sim}(s) \approx 0.1$ we can conclude that the physics simulation forms the computational bottleneck that makes it impossible to perform real-time simulations with the current proof-of-concept implementation. However, in a non-interactive simulation setup, real-time performance is typically of minor importance because the simulated system does not need to be synchronized with a real system.

7.4 Conclusions concerning the validation

Based on a comparison of the scenario bundle-based approach with the conventional approach to include transitions in simulations with multibody software, it could be shown that the proposed approach offers more flexible support for handling multiple scenarios and multiple variations of the artefact/user model. Moreover, it was shown that if applied in particular settings, the new approach has time-saving potential in terms of preparation efforts and simulation time. The particular settings for which time savings can be achieved correspond to assumed needs of designers, namely to investigate (i) scenarios with multiple variations of a design, (ii) complex use processes with multiple transitions, and (iii) complex product models consisting of a large number of modelling elements.

⁶⁵ upper arm, forearm, wrist, palm, 9 phalanges, 81 particles, snack mover, snack door, snack, and button.

Savings in preparation time were expressed by the preparation efforts indicator, which expresses the ratio between the preparation times for the conventional approach and the new approach, respectively. Based on particular assumptions about correspondences between the various kinds of preparation activities, it was shown that preparation time can be saved if scenarios of use with multiple variations of the artefact/user model are being simulated. This is especially the case if more than five model variations are simulated with the same scenario. The time savings increase monotonously but degressively with the number of transitions in the investigated scenario. For large numbers of transitions (ca. 30 or more) the time savings increase near-linearly with the number of investigated model variations.

The reduction of simulation time was expressed with the simulation time indicator, which relates the CPU times needed to perform simulations with the conventional and the new approach. Based on particular (pessimistic) assumptions about computational complexity of logical constructs and simulation models, it was shown that, typically, (i) the scenario bundle-based approach saves simulation time, (ii) the time-saving potential increases monotonously but degressively with the number of movable entities in the simulation model, and (iii) it increases with the number of transitions in the investigated scenario. These findings are *not true* only if the simulation model contains just one or two movable entities. For large numbers of movable entities in the simulation model (ca. 500 or more) it was shown that the time-saving potential increases near-linearly with the number of transitions. This aspect of reducing computation time is of particular interest because such large numbers of movable entities are likely to be present in models with high-resolution particle clouds.

Since the assumptions on which the preparation efforts indicator and the simulation time indicator are based have been phrased in general terms, the metrics for both indicators could be derived independent from the implemented simulation and control algorithms. This means that the above conclusions are not limited to my proof-of-concept implementation that was realized with Adams and Simulink. For a further-developed implementation with a dedicated user interface it may be possible to use more specific and less pessimistic assumptions, leading to a conclusion that even more savings in preparation efforts and simulation time are achievable.

The last aspect that was considered related to simulation time was the feasibility of performing real-time simulations. With the proof-of-concept implementation this was not possible. This is actually not an issue, because the scenario bundle-based approach does not depend on real-time simulation capability because there is no need for human subjects in the simulation loop.

8 CONCLUSIONS AND RECOMMENDATIONS

8.1 Introduction

The research work reported in this thesis aimed to substantiate a new approach that enables designers to simulate the use of products. The activities in the research involved (i) knowledge exploration, (ii) theory development, (iii) proof-of-concept implementation, and (iv) testing, including justification and validation. From the aspect of the conceptualization/design process of products, the approach is supposed to enable designers to identify mismatches and anomalies in use processes straightforwardly. In fact, this has been the ultimate goal of the research efforts and the eventual full-fledged system implementation. The designers may achieve it by conceiving simulations of use processes and exploring problems at early stages.

Subchapter 8.2 concludes about the results of the four main research activities and provides an assessment of what has been achieved in terms of realizing the ultimate goal. Then, in Subchapter 8.3, recommendations are given to guide further research and development towards finalization of the software implementation.

8.2 Conclusions

A comprehensive survey of academic literature and commercially available solutions was conducted, from which it emerged that the functionality offered by existing approaches is insufficient for supporting simulations of typical use processes. Use processes are multifaceted in several respects, and the existing simulation approaches cannot deal with this. The multifaceted nature of use manifests itself with a view to (i) the classes/types of behaviours that have to be predicted, (ii) the large variation that is possible in human users and in surroundings in which a product is used, and (iii) the fact that ongoing manipulative interactions subdivide a typical use process into subsequent episodes in time, enforcing alternating constraints on the simulated behaviours. The diversity of the behaviours in itself has again multiple aspects, which require concurrent consideration. Firstly, behaviours of both humans and artefacts cover several areas of physics (mechanics, thermodynamics, optics, etc.). Secondly, they also manifest behaviours related to control processes that are better described as information processing

based on logic than as signal processing based on the underlying physics. And thirdly, human behaviour involves biological and chemical processes that require other simulation models than the typical physics and information-processing models.

I concluded that the majority of these diverse facets can be addressed by developing a new simulation approach that concurrently enables/supports (i) multiphysics to deal with the diversity in areas of physics, and (ii) logical control to deal with sequences of interactions based on scenarios, and with information processing. Such a simulation approach would leave two issues open for further study, namely, dealing with (i) the large variety of human users and surroundings, which is more of a modelling (or model-instantiation) issue than a simulation issue, and (ii) biological and chemical behaviours of the human body, which are specifically important in use processes related to consumption and fatigue.

As a foundation for the new simulation approach, a three-layered concept has been hypothesized, in which a *behavioural layer* enables computation-based prediction of the behaviour of the models in the *modelling layer* below it, and a *logistics layer*, which includes *scenario bundles*, enables control over the simulation layer. The work in this thesis has focused on the behavioural and logistic layers. The principles underpinning these two layers could be made consistent by developing a dedicated formal theory. In this theory, the two key elements of the concept, that is, (i) multiphysics simulation algorithms that process nucleus-based models in the behavioural layer, and (ii) scenario bundles in the logistics layer, could be unified by following the principles of resource integration, and by referring to a simplified reasoning model of human-artefact interaction.

The theory that enables nucleus-based multiphysics simulation could be developed by building upon ongoing research work in the CADE section. The theory development regarding the behavioural layer focused on connecting simulations and simulation models to the logistics layer. The theory that defines the logistic layer has been developed to comprehensively address all the elements needed to build and execute specifications for the control of simulations. These specifications include scenario bundles in which the designer can specify conjectured interactions. In the theory development, specification elements and terminology could be borrowed from the theory of discrete-event systems and finite automata, with which the logical constructs in the logistics layer have much in common.

To confirm that the theory of resource-integrated simulation controlled by scenario bundles could be converted into a structured set of processable algorithms and to demonstrate it by applying it to sample cases, a proof-of-concept implementation was built. In order to realize this implementation within the constrained time and resources that were available, it was decided not to build a dedicated full-fledged modelling and simulation system. The two most important sacrifices that were made were (i) using commercially available, configurable software as a proxy for dedicated software developed from scratch, and (ii) using drastically simplified human models as proxies for models founded on a scientific basis of human anatomy and low-level motion control behaviour.

Regarding the first proxy it turned out that a combination of the commercial software packages MSC Adams and Matlab/Simulink could be deployed as proxies without compromising the testing objectives. Two disadvantages directly followed from the choice for these software proxies. Firstly, several opportunities that the theory offers for automation of modelling and specification activities could not be implemented. Hence, longer preparation times for experimental models and specifications had to be taken for granted. Secondly, modelling concepts that had been formally defined based upon the principles of resource-integrated modelling had to be translated to similar concepts that were available in Adams and Simulink.

Due to the second proxy it was not possible to obtain realistically simulated human motion patterns. A motion pattern can be seen as a shift from one body posture to another one. In specifying proxy models of low-level human motion control, the priority has been that, starting from a given initial posture, the required end posture was achieved without paying attention to achieving realism in the intermediate postures (or in the in-between trajectories of points of the human body). For instance, for a 'reaching' motion pattern, the proxy for low-level control would be aimed at arriving at the desired end point of reaching, but no validated models were used to obtain realistic motions between start and end of reaching. The fact that some of the simulated motions conveyed a somewhat 'robotic' impression was taken for granted.

To show that the developed theory is consistent and feasible and to demonstrate the novel functionality that conventional simulations cannot offer, the proof-of-concept implementation was tested by applying it to a series of sample cases, applying the principle of component-based application development. In the initial four sample cases scenario-controlled simulations of elementary interactions have been carried out and tested, which were then combined as components into simulations of three composite use processes. The third composite case, in which the use of a snack dispenser was simulated, was tested with a version of the proof-of-the implementation that included all the aspects and details of the theory as presented in this thesis.

By running simulations of using the snack dispenser, the novel functionality could be shown by (i) simulating connected combinations of basic physical interactions (scenarios) (ii) simulating multiple scenarios from one bundle, and (iii) repeating simulations with variations of the product (and its human user) to obtain feedback about the product design. The case studies also showed that modelling flexible bodies based on the nucleus principle made it possible to simulate human grasping, which is important in typical interactions between humans and artefacts.

Testing the proof-of-concept implementation revealed that the physics simulation is susceptible to small changes in event-threshold values that are used to specify transitions between interactions. The proxy model of low-level human motion control seemed to be the most likely cause. Particular control instructions in this model involved small discrete motion corrections that could occur repeatedly at a near-constant frequency, thus unintentionally causing spring-damper

elements to resonate. In some cases, this made computations result in extreme local forces and accelerations, causing unstable or crashing simulations. This issue was not further investigated. During testing it could be resolved by fine-tuning event-threshold values by trial and error, which was however time-consuming.

To justify the theory, its properness was investigated and the limits of its applicability were identified based on reasoning with the consequences of the included concepts. The investigations identified two main categories of limitations, and further subcategories comprising typical application cases for which the theory fails to support the assumptions of the hypotheses. These categories are: (i) lack of support to fulfil particular needs of the designer (for instance, if he wants to change models *during* a simulation) and (ii) lack of support for products and use processes that have characteristics to which the theory does not apply (for instance, use processes that depend on learning). It has to be mentioned however, that so far the inventory of limitations has not been exhaustive.

The theory and the approach were validated by comparison with a reference approach that would offer designers the possibility to interconnect simulations by using a conventional simulation system. This could be achieved by specifying a workaround based on step functions. However, argumentation based on facts showed that this conventional reference approach cannot match the functionality that the scenario bundle-based approach offers to handle multiple scenarios and model variations efficiently.

The novel functional affordances are only useful for designers if they do not need more preparation efforts and if simulation runs do not take more computation time. These aspects could be validated by defining indices based on general assumptions (i) about correspondences between the various kinds of preparation activities and (ii) about computational complexity of logical constructs and simulation models. Even though the assumptions had been formulated in a pessimistic sense (giving all possible advantage to the conventional reference approach), it could be shown that:

- the scenario bundle-based approach saves preparation efforts if five or more variations of models (of the product, the user, and/or the environment) are investigated by simulation with the same scenario.
- the relative advantage in preparation efforts increases with the number of investigated model variations and with the number of transitions between interactions in the investigated use process. If the number of transitions is sufficiently large (ca. 30 or more), the increase as a function of the number of model variations is near-linear.
- simulations with the scenario bundle-based approach requires less total CPU time if the number of moving entities (parts, particles) in the simulation model is two or more.
- the relative advantage in processing time increases with the number of transitions and the number of moving entities. If the number of moving entities is sufficiently large (ca. 500 or more), the increase as a function of the number transitions is near-linear.

The reason for these advantages is that in a scenario bundle, all transitions in the use process can be predefined in advance and independent of the actual simulation results, whereas the conventional approach requires transitions to be defined at prescribed points in time, which can only be found by running a series of partial simulation runs (one run for each transition).

Wrapping up the findings of the research, it can be concluded that designers will benefit from the formal means that logical models offer to capture knowledge about anticipated (intended or unintended) scenarios of decision-making by human users, and about the way products are programmed to process information. If, ultimately, the method and tool are made available to designers in a full-fledged form, I expect that further additional benefits can be realized in the design process, such as: (i) reduced need for deployment of human subjects in user testing, (ii) support of systematic consideration of human decision-making by designers, and (iii) increased efficiency by reuse⁶⁶ of simulation resources that describe anticipated usage in an unambiguous form.

8.3 Recommendations for future research and development

It is to be expected that several of the shortcomings, limitations, and inconveniences encountered during the research work can be resolved by developing a dedicated system. In further research and development efforts towards a design support system for simulation of use processes, four main directions can be distinguished towards the following objectives: (i) the realization of a nucleus-based modelling and simulation system, (ii) the realization of a system for specification, modelling, and execution of constructs in the logistics layer, (iii) the realization of a repository of comprehensive human models, and (iv) realizing an integrated system for modelling, specification, and simulation of humans, artefacts, and human-artefact interaction. These future efforts have to deal with several mutually dependent research and development challenges. Below I have outlined the future activities for each of the four main directions based on the current state of affairs.

One of the most challenging tasks is the development of a dedicated environment for nucleus-based modelling that will also support simulation of physical behaviour outside the mechanical domain (thermodynamics, acoustics, etc.) and that involves dedicated algorithms to make fast, preferably real-time, simulations possible with high-resolution particle-based models. In the mechanical domain two points of attention are (i) to achieve realistic simulation of deformation behaviour of particle-based models by finding appropriate 3D connection patterns and appropriate relations to define the connecting nuclei and (ii) to find a

⁶⁶ Actually, the reuse of simulation resources appears on two levels. Firstly, within one product development project, scenario bundles (and other logical constructs) can be reused across variations and evolutions of models. Secondly, across multiple projects, partial scenarios can be reused for products involving similar use interactions.

way of modelling surfaces that enables correct simulation of contacts (i.e., collision and friction). In order to facilitate the specification of control relations, it is recommended to consider allowing prescribed motions beside prescribed forces and torques. This would eliminate the need for inserting pid/pi/pd controllers between control and simulation, as is required by the conventional forward-dynamics simulation algorithms.

To operationalize nucleus-based modelling and simulation, either as a stand-alone software package or in combination with logical control, a user interface for the creation of simulation models must also be developed. Such a user interface can either be dedicated, to enable creation of nucleus-based models from scratch and support of incomplete conceptual models, or it can be aimed at importing and converting models from conventional cad systems. In both cases, algorithms are needed to automatically discretize components into particle clouds with connecting nuclei between the particles.

In the further development of a subsystem for specification, modelling, and execution of constructs in the logistics layer, the first recommended step is to investigate the extent to which product designers can familiarize themselves with statecharts, or with alternative ways of building logical specifications. The user interface of Simulink Stateflow seems to be reasonably mature, and it might be worthwhile to investigate whether it forms an acceptable means of specification for designers.

Once the logical models of low-level control of human motions have reached a reasonably mature stage of development, it should also be investigated how these models can be managed and combined with scenario bundles. It should also be clarified to what level of detail designers are able to specify interactions, and how the user interface of the logistics layer should offer specification elements for them. With low-level human motion control models as elementary building blocks, it may be possible to prepare modular logical sub-routines of human decision-making and muscle control that can be reused in different simulations.

To achieve further integration of the modelling and specification resources it might be worthwhile to investigate possibilities to express finite automata as differential equations. By implementing principles proposed by Branicky [270], it would be possible to represent both the logistics layer and the simulation model as continuous systems, which might increase computational efficiency during simulations.

Further elaboration of human models seems to be of the same complexity as the realization of a nucleus-based modelling and simulation system. The challenge to enable creation of realistic and complete human models involves a multidisciplinary effort unifying aspects of (bio)mechanics (possibly in combination with biochemistry), control, and anthropometrics.

To make mechanical simulations of the human body sufficiently realistic, it may be needed to include knowledge about the nonlinear deformation behaviours of the various human-body tissues, and to find ways to include this knowledge into relations within particle clouds representing these tissues. However, part of the

internal physical behaviour of the human body possibly does not directly influence interaction processes. To reduce computational complexity it is worthwhile to investigate how these behaviours can be bypassed in, or excluded from, simulations.

Another demanding task related to the development of human models is the inclusion of more detail knowledge about low-level control behaviour in the human interaction construct. However, some researchers have recently claimed that they have resolved the problem of simulating low-level control of human motions [e.g., 191,243]. Adopting these solutions in future work is worth considering. Alternative solutions can be considered as well. For instance, if available from research work in human motor science, invariants of motion patterns may form an attractive starting point. Alternatively, if generalization and parameterization of recorded motions is possible, patterns obtained from motion-capture based observations can be considered. To enable testing of virtual products with virtual users of various size, age and gender, anthropometrics knowledge should be operationalized for instantiation of human models corresponding to different anthropometric characteristics from a database.

The integration of the above subsystems involves two main aspects: (i) linking models and specifications and (ii) providing the designer an interface to perform controlled simulations. In the proof-of-concept implementation the models were linked manually. However the theory offers extensive opportunities for automation of these activities, since all connecting signals (meter signals, control signals and events) appear in at least two constructs. Therefore, the aim should be to eliminate linking of models as a separate activity. If, for instance, the designer specifies a meter event in the scenario bundle, a list can be presented of all meter values already defined in the simulation model. By selecting values from the list, and defining the event-generating function, the event orientation, and the threshold, the connection should be possible without using a specific linking specification interface. If in the same case a new meter value is needed that is not yet in the list, it should be possible to specify it on-the-fly in the simulation model.

The interface functionality for running and evaluating controlled simulations was very limited by the proof-of-concept implementation. In a full-fledged system, it should be possible to view concurrent animations of both the physics simulation and the succession of states and transitions in logical construct. Both animations should be available during as well as after a simulation.

One other issue related to integration that calls for a solution (to be realized either in the nucleus-based simulation system or in the logistics layer) is the influence of logical control instructions on the occurrence of resonance in spring-damper elements. In the proof-of-concept implementation this problem caused instabilities in simulation computations. Possible solutions are (i) to eliminate sudden direction changes, either by implementing continuous control algorithms for small repetitive corrections in prescribed motions or by tweaking settings of PI/PD/PID controllers that calculate forces from prescribed motions, or (ii) to apply a workaround in order to eliminate higher eigenfrequencies from simulation

computations. [e.g., 271]. To obtain more insight into what goes wrong, a first step might be to enhance the current simulation constructs with debugging elements as was done by Filla [272], in an Adams simulation that was also controlled by Simulink Stateflow.

The above order of presenting the activities does not indicate priorities or sequencing. Most likely the further development will show parallel progress in achieving all the four objectives. Apart from system development activities and research directly related to system development, it is also necessary to justify and validate the achievements in a more comprehensive way than was possible with the proof-of-concept implementation. To that end, scheduling of the development activities should take into account the necessity of the following justification and validation tasks:

- Evaluation of the utility of the system by trials with designers;
- Comprehensive testing with a large number of case studies;
- More accurate evaluation of the preparation time and simulation time by carrying out experiments with designers working on case studies;
- Further systematic investigation to identify limitations of the theory that have not been identified in the justification;
- Justification of the realism of the human motion patterns that the system generates by conducting empirical comparisons involving human subjects.

SUMMARY

An important goal of user-centred design is designing consumer durables for optimal interaction with users, i.e., considering how they can be used by various users in various circumstances. Possible forms of use can be investigated with human subjects and physical prototypes, but recruitment of subjects and organization of test sessions are typically time and money-consuming. In the case of conceptual design, adequate physical prototypes are often unavailable, which makes testing even more problematic. Under these circumstances, simulations with virtual prototypes have proven an attractive and often efficient solution. Simulations of complex use processes however, are not supported sufficiently by currently available simulation methods and tools. This is caused by three problems, which form the starting point of my research.

The first research problem is that in order to prepare simulations of use processes, designers have to deal with (or choose from) many types of models, because use processes involve different behaviours (e.g., mechanical, thermal, biological) which require different simulation approaches. These approaches involve a multitude of models and data constructs. The second problem is that conventional simulation approaches do not offer a means to consider use as a *sequence* or network of connected interactions as it happens in reality. Although they may foresee several of these interactions as coherent sessions of use, designers are restricted to investigate interactions one at a time. The third problem is that human interaction in use processes depends on human decision-making (performed by the brain) for which no practicable simulation methods exist.

In order to solve these problems I identified the following questions:

- (i) How can multiple areas of physics be simulated concurrently?
- (ii) Is it possible to simulate arbitrary behaviours using just one generic modelling principle for entities (artefacts) and their behaviours?
- (iii) How can designers use their conjectures about interaction processes to:
 - (a) specify changes in simulation input data over time?
 - (b) devise specifications to substitute simulation of the reasoning and decision-making in human users that leads to these changes?
- (iv) How can the models and specifications needed to simulate use processes be connected and arranged in a way that enables computational processing, and at the same time allows reasoning about human-product interactions?
- (v) How can a concept of a system to support use-process simulations be organized to facilitate modular development of building blocks supporting distinctive design activities?

Knowledge exploration. To address these questions, first the available knowledge had to be explored regarding (i) existing simulation approaches, i.e., the processes and behaviours they cover and the models and constructs they use, and (ii) human control of interactions, i.e., how does human decision-making relate to the physical interactions taking place in the human-artefact system and

how does it relate to interactions typically foreseen by designers? This knowledge exploration was conducted in the form of a literature search, extended with an inventory of commercial simulation packages.

The findings of the knowledge exploration were as follows. Regarding processes and behaviours related to use, existing simulation approaches differ (i) in their capacities to bring together the various areas of physics in behavioural simulation, and (ii) in their potential to consider interacting humans and artefacts. In the first respect, discretized-model based simulations, such as finite-element and particle based ones, form the state of the art. In the second respect, however, simulations based on volumetric models prevail. The physics covered by these approaches is however typically limited to rigid-body mechanics.

The literature on human interaction control distinguishes high-level and low-level control mechanisms. Decision-making forms the highest level. It is based on input from perception, and it results in instructions for body movements. These instructions are further processed by low-level control mechanisms, which produce contraction-regulating signals for muscles.

In the literature on designing interactive systems, especially in literature on human-computer interaction, various approaches for designers to specify foreseen interactions are addressed. In these approaches, *scenarios* (or use cases) play a central role as carriers of foreseen interactions. A scenario of use is a possible way for a human user to control his interactions with a given product in given surroundings. It represents control of interactions at the level of decision-making, usually by depicting decisions as logical junctions connecting alternative paths through the use process. According to the literature, multiple scenarios are possible for the use of one product, and it has been suggested that organized sets of scenarios offer designers the expression power to specify all foreseen use. Hence, my third and fourth research questions can be combined and reformulated as: *“Can an approach prescribing the use of such modelling principle offer control mechanisms to simulate use as one holistic process, based on designer-conjectured organized sets of use scenarios?”*

To operationalize scenarios for quantitative simulations, the logic connecting foreseen decision points has to be specified in some formal, unambiguous way. The literature favours graphical logical representations, such as (dialects of) state transition diagrams, Petri nets, and statecharts, because they are easier to comprehend than language-based and algebra-based representations. Of these representations, statecharts offer the highest representation potential without the need to resort to a particular dialect.

Theory and concepts of resource-integrated interaction simulation. To derive a testable theory that provides answers to the research questions and would serve as the basis of any practical solution, I hypothesized that (i) the concept of *resource integration* can be deployed to organize and connect sequences of interactions.

Resource-integration builds upon the principles of *nucleus-based modelling*, which were developed in ongoing research within the CADE section. By focusing on relations between entities rather than on the entities, nucleus-based model-

ling allows simulation of multiple types of behaviour ('multiphysics'). In order to enable simulation of interconnected interactions, resource integration adds *transition relations* between subsequent situations in time and it provides means to arrange situations in a logical network. The hypothesis specifically addresses this enhancement to nucleus-based modelling offered by resource-integrated modelling. No hypothesis was dedicated to the potential offered by nucleus-based models to simulate multiphysics. In the further elaboration, this aspect was included for demonstrative and illustrative purposes but not assessed through validation and justification.

Furthermore, I hypothesized that (ii) transition relations can be used to specify human interactions in *scenario bundles*, i.e., designer-conjectured organized sets of scenarios.

Finally, I hypothesized that, (iii) to bring the aforementioned concepts together into a use-process simulation approach, the simplified reasoning model of human-artefact interaction shown in Figure 55 can be used. In accordance with the findings of the knowledge exploration, this reasoning model arranges and connects typical functions performed by humans and artefacts. These are shown as connected white blocks. The shaded blocks depict the models and specifications involved in use-process simulation. The *physics simulation model* represents the part of interaction simulated as physics (which is, in this thesis, limited to mechanics). It is modelled with 'conventional' nucleus elements – i.e., it contains no transition relations. The other models and specifications represent control processes, of which the governing physical phenomena are interpreted as information flows. The relations in these *logical constructs* are transition relations.

There are two logical constructs that represent mechanisms controlling human behaviour. The *scenario bundle* is the logical construct in which the designer specifies conjectured interactions. The *model of low-level logical control of human motion* is built up from predefined behavioural modelling elements based on knowledge from human motor science. Together the scenario bundle and the model of low-level logical control of human motion control simulated interactions through the transition relations they contain. The optional *procedure structure* is a formal specification similar to the constructs describing human control.

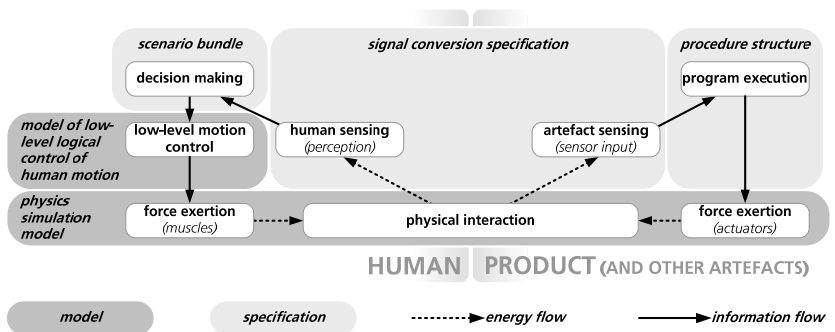


Figure 55. Simplified reasoning model of human-product interaction, and arrangement of models and specifications in scenario bundle-based simulation

If needed, the designer can use it to specify the procedures applied by embedded software in products to control actuators based on input from sensors.

To investigate variations of use, the designer can do what-if type simulations, either by varying the content of the scenario bundle or by varying the model of the human user and the model of the product, of which the procedure structure is a part.

Proof-of-concept implementation. A proof-of-concept system was developed and implemented to the level of experimental testability to support justification and validation of the hypothesized approach by showing that the developed theory is consistent and feasible. From the aspect of testing, my objective has been (i) to show that my approach lends itself to new functional affordances not available in current commercialized systems and (ii) to show the utility of the implemented system and to assess the convenience of use from technical aspects. A third objective would have been to encounter the benefits of application in design processes. However, I found that the current early stage of development allowed only a qualitative indication of possible benefits rather than a thorough assessment.

With a view to the above objectives and preferences, developing a full-fledged system with a dedicated user interface to support creation of constructs and computation of use simulations was not needed. Instead, my efforts focused on realization and combination of the following sub-functions: (i) resource-integrated modelling and simulation (i.e., adding transition relations to nucleus-based modelling and simulation), and (ii) simulation control based on scenario bundles.

Resource-integrated modelling and simulation presupposes nucleus-based modelling and simulation, for which no dedicated software components were available. Rather than developing software components for all sub-functions, I chose to rely on proxies offering similar functionality. To emulate simulatable nucleus-based models, the multibody dynamics package MSC Adams was used. Transition relations were specified and modelled with Matlab Simulink as state transitions using the statechart notation. Simulink was also used to execute the transitions and thus control the physics simulation in Adams.

Justification. To assess the veracity and the credibility of the theory as well as its implications in applications, I followed the approach of *scoping its properness by reasoning about its consequences*. For each hypothesis the corresponding claims of the theory have been investigated by considering those conditions that incapacitate it from maintaining the assumed truth. The result of scoping could convincingly be expressed by categorizing typical application cases for which the theory fails to support the assumptions of the hypotheses.

Reasoning with the implications of the theories revealed limitations belonging to two main categories: (i) possible needs of designers for which the theory does not offer a solution, and (ii) classes of products and typical uses to which the theory does not apply. A limitation of the first category is, for instance, lack of support for settings in which designers want to change models while a simulation is running. The reason is that such changes remove the assumed interrela-

tions between consecutive interactions. A limitation of the second category is, for instance, that it is not possible to control and simulate use processes governed by learning based on past experience. The reason is that learning involves human decision-making across consecutive use processes, which cannot be represented by state transitions.

The inventory of limitations has not been exhaustive. Further systematic investigation may point to other, yet uncharted limitations.

Validation. The proposed approach and its proof-of-concept implementation were tested with respect to the following aspects: (i) showing that the approach lends itself to the hypothesized new functional affordances, and (ii) to show the utility of the implemented system by addressing the convenience of use from technical aspects.

For that purpose I applied the approach and pilot system to a series of sample products (i) of which the use involves multiple consecutive interactions, and (ii) for which various interaction sequences are possible. To demonstrate the multiphysics capacities of nucleus-based models, it was additionally required that (iii) the use process depends on a combination of physical phenomena that cannot be simulated with existing approaches.

The last and most comprehensively discussed sample product was a snack dispenser. Its use process involved (i) a sequence of reaching, pushing and releasing a button, grasping an object, lifting it, and carrying it, and (ii) variations in interaction sequences, which can be effected by playing with the specification of human hesitation in the scenario bundle. Additionally, (iii) the use process depends on rigid-body dynamics in combination with large deformations (in human grasping).

As a reference benchmark to compare the approach and pilot implementation to, a procedure was deduced for employing conventional multibody simulation software in order to achieve the closest possible match to the hypothesized new functionality.

Regarding evaluation of novel functional affordances, the demonstrative example showed that the proposed logical control mechanism made it possible to design various use-process scenarios and to monitor the conduct of these scenarios. Contrary to the benchmark procedure based on conventional multibody simulation, the control mechanism can be used to (i) simulate multiple scenarios of concatenated interactions based on one scenario bundle and (ii) control the physical simulation in the case of moderately varied artefact geometry and/or characteristics of the human by the same scenario bundle.

The utility of the system in terms of convenience of use was objectified by defining two quantitative indicators, one for the required efforts to prepare simulations and one for the simulation time.

The preparation efforts indicator (PEI) expresses preparation efforts in terms of the total number of required activities by the designer as a ratio between what is needed for the scenario-bundle based approach and what is needed for the benchmark procedure. Using this indicator it was shown that preparation time can be saved if scenarios of use with multiple (at least five) variations of the

artefact/user model are being simulated. The time savings increase with the number of transitions in the investigated scenario.

The simulation time indicator (STI) expresses the total CPU time as a ratio between what is needed for the scenario-bundle based approach and for the benchmark procedure, which turns out to be a function of (i) the CPU time needed for physics simulation of a given investigated use process, (ii) the time needed for execution of the control instructions for that process in the scenario bundle and procedure structure, (iii) the number of transitions in that process, and (iv) the number of moving entities (e.g., parts) in the human-artefact system. It was shown that simulations based on scenario bundles are faster if the number moving parts is greater than or equal to two. The advantage increases with a further increase of the number of parts, and with an increase of the number of transitions. For the abovementioned typical use process of the sample product, the value of STI was shown to be approximately equal to 10, i.e., the scenario-bundle based approach reduced CPU time by 90% compared to the benchmark procedure.

Discussion. Reflecting on the employed research methods, the choice to use commercial systems as proxies for realizing a proof of concept implementation was pointed out as the most critical impact factor on the conduct of the research work. The main positive consequence was, that a proof of concept could be realized within the timeframe of a PhD project carried out by one person with a convincing and reasonably realistic validation.

There were two important negative consequence of this choice. The first was susceptibility of the physics simulation to small changes in models and transition conditions. This is caused by (i) the drastically simplified human model, which had to be tolerated in the absence of high-fidelity low-level human control models and (ii) the complicated workarounds that had to be adopted in order to build particle-based models in Adams, which allowed me to use this rigid-body simulation system to simulate flexible bodies. Because of this susceptibility to small changes, models and specifications required laborious fine-tuning in order to obtain reliable simulations of complete use processes. Consequentially, there was not enough time to elaborate other sample products and their use processes. A reason to study a larger number of cases involving various sample products (preferably in real-life design projects) would be to build a stronger case for scenario-bundle based simulation through increasing statistical significance. There is, however, uncertainty about what statistical significance would be needed and how it can be achieved.

The second negative consequence was that no dedicated user interface was available for testing the approach by trials with designers in practice.

Conclusions. To simulate use processes, the concept of resource integration can be deployed to organize and connect sequences of interactions. It allows the designer to use transition relations for specifying human interactions in scenario bundles, i.e., conjectured organized sets of scenarios, based on a simplified reasoning model of human-artefact interaction. A proof of concept implementation built with commercialized software could be operationalized to show that my

approach could be successfully applied to a sample product to (i) simulate connected combinations of basic physical interactions (scenarios), (ii) simulate multiple scenarios from one bundle, and (iii) simulate variations of the product (as well as its user and use environment) to obtain feedback about the product design. In current commercialized systems, these new functional affordances are not available. From the aspects of preparation efforts and simulation time, the convenience of use appears to be acceptable when investigating complex use processes with various transitions between interactions. It has to be mentioned though, that simulation of processes in which transitions are of no significance was not the application area where my approach was intended to compete with conventional approaches.

To complete the PhD research in a reasonable time, some important issues had to be left open for future investigation and development. I recommend the following activities to be part of follow-up research: (i) realization of a dedicated system with a dedicated user interface that can be tested with designers, (ii) inclusion of more detail knowledge about human motor-control behaviour, and (iii) investigation of possibilities to reduce preparation efforts by offering automation and predefined templates.

SAMENVATTING

Als men tijdens het ontwerpen van producten rekening wil houden met toekomstige gebruikers, is optimale interactie een belangrijk aandachtspunt. Daarbij gaat het erom, hoe producten door verschillende gebruikers onder verschillende omstandigheden kunnen worden gebruikt. Hoewel mogelijke vormen van gebruik kunnen worden onderzocht met proefpersonen en tastbare prototypes, is het werven van proefpersonen en het organiseren van testsessies kostbaar en tijdrovend. Als het gaat om conceptontwerp zijn geschikte fysieke (tastbare) prototypes vaak niet beschikbaar, wat het testen nog problematischer maakt, en dan biedt computersimulatie met virtuele prototypes vaak uitkomst. Met de huidige simulatiemethodes en -tools zijn simulaties van complexe gebruiksprocessen echter nog niet goed mogelijk. Daaraan liggen drie problemen ten grondslag, die als uitgangspunt van mijn onderzoek dienden.

Het eerste onderzoeksprobleem is dat, om simulaties van gebruiksprocessen op te zetten, ontwerpers te maken krijgen met – of moeten kiezen uit – vele verschillende soorten modellen, omdat gebruiksprocessen verschillende vormen van gedrag omvatten (bijvoorbeeld mechanisch, thermisch of biologisch bepaald gedrag) die elk een andere simulatieaanpak vergen. Het tweede probleem is dat gangbare simulatiebenaderingen geen mogelijkheid bieden om gebruikssessies te beschouwen als de reeksen of netwerken van samenhangende interacties die ze in werkelijkheid zijn. Hoewel ontwerpers misschien wel al een beeld hebben van samengestelde interacties die kunnen plaatsvinden, kunnen ze in simulaties deze interacties slechts één voor één bestuderen. Het derde probleem is dat de gebruiksinteractie afhangt van menselijke besluitvorming (door de hersenen), een proces waarvoor nog geen praktisch toepasbare simulatiemethodes bestaan.

Om deze problemen aan te pakken heb ik de volgende onderzoeksvragen opgesteld:

- (i) Hoe kunnen gedragsvormen die onder verschillende deelgebieden van de fysica vallen gelijktijdig worden gesimuleerd?
- (ii) Is het mogelijk om willekeurige gedragsvormen van objecten te simuleren door uit te gaan van een generiek modelleerprincipe?
- (iii) Hoe kunnen ontwerpers hun veronderstellingen over mogelijke interactieprocessen benutten om
 - (a) over de tijd veranderende gegevensinvoer te specificeren voor simulaties?
 - (b) procesbeschrijvingen op te stellen die kunnen worden gebruikt als een substituut voor het simuleren van redeneren en beslissen door menselijke gebruikers?
- (iv) Hoe kunnen de modellen en de specificaties die nodig zijn voor het simuleren van gebruiksprocessen met elkaar worden verbonden en zodanig geordend dat ze door computers verwerkt kunnen worden en tegelijk door de ontwerper kunnen worden toegepast om over mens-productinteracties te

redeneren?

- (v) Om de bovenstaande kwesties onafhankelijk te kunnen onderzoeken moet een conceptstelsel voor simulatie van gebruiksprocessen ontwikkeld worden. Een dergelijk stelsel is opgebouwd uit modules die de verschillende gerelateerde ontwerphandelingen ondersteunen. Hoe kunnen deze modules zodanig worden gekozen dat de delen die beslissend zijn m.b.t. de bovenstaande hypothesen onafhankelijk ontwikkeld en getoetst kunnen worden?

Kennisexploratie. Om op deze vragen te kunnen ingaan werd eerst een kennisexploratie uitgevoerd om inzicht te verkrijgen in (i) bestaande simulatiemethoden, d.w.z. de processen en gedragsvormen waarvoor ze kunnen worden gebruikt, en (ii) de wijze waarop mensen interacties beheersen; d.w.z. hoe verhoudt zich menselijke besluitvorming tot de fysieke interacties in het mens-productstelsel en tot hoe kunnen interacties die ontwerpers gewoonlijk voorzien daarin een plaats krijgen? Deze kennisexploratie werd uitgevoerd door een literatuurstudie te verrichten en een inventarisatie te maken van commercieel verkrijgbare simulatiesoftware.

De uitkomsten van de kennisexploratie waren als volgt. Met betrekking tot gebruiksgelateerde processen en gedragsvormen verschillen bestaande simulatiemethoden (i) in hun vermogen om de verschillende deelgebieden van de fysica in simulaties te verenigen, en (ii) in hun toepasbaarheid op interactieprocessen tussen mens en product. Als we kijken naar het eerste aspect, dan zijn simulatiemethoden die gebaseerd zijn op gediscretiseerde modellen (bijvoorbeeld d.m.v. eindige elementen of partikels) het verst ontwikkeld. Met betrekking tot het tweede aspect bevinden zich echter simulaties met volumetrische modellen in de voorhoede. Fysisch gedrag is in de laatstgenoemde methoden echter beperkt tot de mechanica van starre lichamen.

De literatuur over menselijke interactiebeheersing maakt onderscheid tussen beheersing op hoog en laag niveau. Besluitvorming die resulteert in bewegingsinstructies voor lichaamsdelen op basis van waarneming, vormt het hoogste niveau. Op laag niveau worden de bewegingsinstructies omgezet in signalen die spiercontracties regelen.

De literatuur over het ontwerpen van interactieve systemen, en met name de literatuur over mens-computerinteractie, behandelt verscheidene methoden waarmee ontwerpers veronderstelde interacties kunnen specificeren. In deze methoden spelen *scenario's* (ook wel *use cases*) als middel om interacties vast te leggen een centrale rol. Een gebruiksscenario beschrijft een mogelijk verloop van de interacties die een mens aangaat met een gegeven product in een bepaalde omgeving. Het vertegenwoordigt interactiebeheersing op het niveau van besluitvorming, gewoonlijk door beslissingen weer te geven als logische knooppunten van waaruit verschillende 'paden' vertrekken waarlangs het gebruiksproces kan verlopen. Volgens de literatuur zijn er voor het gebruik van een bepaald product steeds verscheidene scenario's denkbaar, en er is gesuggereerd dat geordende stelsels van scenario's de uitdrukingskracht bieden om alle voorziene vormen van gebruik vast te leggen. Op basis daarvan konden mijn derde en vierde onderzoeksvraag worden gecombineerd en geherformuleerd als "*Kan een methode*

die het gebruik van een dergelijk modelleerprincipe voorschrijft beheersingsmechanismen bieden waarmee gebruik kan worden gesimuleerd als één holistisch proces, uitgaand van door de ontwerper bedachte geordende stelsels van gebruiksscenario's?"

Om scenario's operationeel te maken voor kwantitatieve simulaties moet de logica die de voorziene beslissingspunten verbindt formeel en ondubbelzinnig worden vastgelegd. Volgens literatuur verdienen grafische logische voorstellingswijzen, zoals toestandsdiagrammen, Petrinetten en statecharts daarbij de voorkeur. Als men het gebruik van dialecten van deze representaties wil mijden bieden statecharts de meeste uitdrukingskraacht.

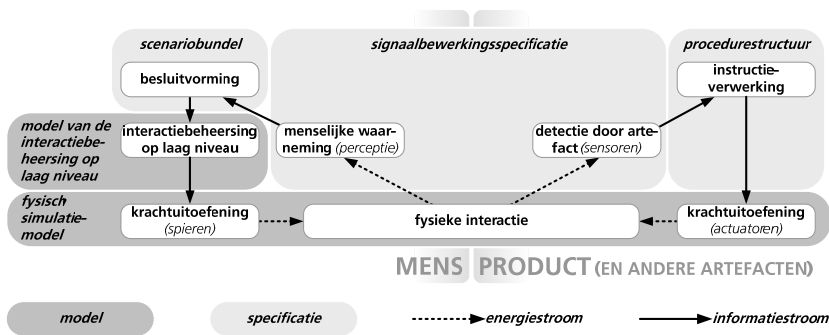
Grondslagen van interactiesimulatie met geharmoniseerde kennisbronnen. Om een toetsbare theorie af te leiden die de antwoorden geeft op de onderzoeksvragen, en die kon dienen als basis voor een praktische oplossing, stelde ik als hypothese dat (i) het beginsel van *harmonisatie van kennisbronnen* kan worden toegepast om interactiesequenties te ordenen en te verbinden. Dit beginsel bouwt voort op de principes van *nucleusgebaseerd modelleren*, die zijn ontwikkeld in het kader van lopend onderzoek in de sectie CADE*. Door de nadruk te leggen op relaties tussen entiteiten in plaats van op de entiteiten zelf, kunnen met nucleusgebaseerde modellen simulaties worden uitgevoerd die verscheidene deelgebieden van de fysica omvatten ('multifysica').

Harmonisatie van kennisbronnen leidde ertoe dat er aan het nucleusgebaseerd modelleren *transitiërelaties* tussen opeenvolgende situaties werden toegevoegd, zodat samenhangende interacties kunnen worden gesimuleerd. De eerste hypothese gaat met name in op deze toepassingsmogelijkheid die geharmoniseerde kennisbronnen toevoegen aan nucleusgebaseerd modelleren. Er was geen hypothese gewijd aan de multifysica-simulatiecapaciteiten die nucleusgebaseerd modelleren biedt. In de verdere uitwerking is dit wel meegenomen voor demonstratieve en illustratieve doeleinden, maar het is niet gevalideerd en geverifieerd.

Voorts heb ik als hypothese gesteld dat (ii) transitierelaties kunnen worden gebruikt om menselijke interacties te specificeren in *scenariobundels*, d.w.z. geordende stelsels van door de ontwerper bedachte scenario's.

Tenslotte voerde ik als hypothese op dat (iii) om de voor genoemde begrippen samen te brengen een vereenvoudigd redeneermodel kan worden gebruikt dat is weergegeven in Figuur 1. Conform de bevindingen van de kennisexploratie ordent en verbindt dit model de karakteristieke functies die mensen en producten vervullen. In de figuur zijn de functies weergegeven als witte blokken, terwijl de grijze blokken de modellen en specificaties voorstellen die gehanteerd worden voor simulatie van gebruiksprocessen. Het *fysische simulatiemodel* vertegenwoordigt dat deel van de interactie dat wordt gesimuleerd op basis van de fysica (die in het kader van dit proefschrift beperkt is gebleven tot mechanica). Het wordt gemodelleerd m.b.v. 'conventionele' nucleuselementen, d.w.z. het bevat

* *Computer-aided design engineering*: computerondersteund construeren



Figuur 1. Vereenvoudigd redeneermodel ter beschrijving van mens-productinteractie, en ordening van modellen en specificaties in het simuleren met scenariobundels

geen transitierelaties. De andere modellen en specificaties vertegenwoordigen beheersings- en regelprocessen, waarbij de betrokken fysische processen worden geïnterpreteerd als informatiestromen. De relaties in deze *logische constructen* zijn transitierelaties.

Twee logische constructen vertegenwoordigen de beheersings- en regelmechanismen van menselijk interactiegedrag. De *scenariobundel* is het logische construct waarin de ontwerper de voorziene interactiehandelingen specificeert. Het *model van de interactiebeheersing op laag niveau* is een logisch construct gebaseerd op voorgedefinieerde modelleerelementen die gedrag beschrijven op basis van kennis over menselijke motoriek. De *scenariobundel* en het *model van de interactiebeheersing op laag niveau* sturen de te simuleren interacties aan door de transitierelaties die ze bevatten. De facultatieve *procedurestructuur* is een formele specificatie die gelijksoortig is aan de constructen voor menselijke interactiebeheersing. Zo nodig kan de ontwerper deze gebruiken om de procedures te specificeren waarmee ingebouwde software is geprogrammeerd om actuatoren aan te sturen op basis van input vanuit sensoren.

Om variaties op gebruiksprocessen te bestuderen kan de ontwerper 'wat-als?'-simulaties uitvoeren door veranderingen door te voeren in de modellen van de menselijke gebruiker en van het product (inclusief de *procedurestructuur*).

Pilotimplementatie. Een pilotimplementatie werd ontwikkeld en zover uitgewerkt dat de voorgestelde methode experimenteel kon worden beproefd om aan te tonen dat de ontwikkelde theorie consistent en werkbaar is, en om langs deze weg verificatie en validatie mogelijk te maken. Wat betreft het beproevingsaspect was mijn doel (i) te tonen dat mijn methode nieuwe functionaliteit biedt die in bestaande systemen niet aanwezig is en (ii) de bruikbaarheid van geïmplementeerde systeem te tonen en het gebruiksgemak ervan te beoordelen op technische criteria. Een derde doel zou zijn geweest om de voordelen bij toepassing in de praktijk van het ontwerpproces aan te tonen. Ik moest echter concluderen dat de huidige stand van ontwikkeling geen grondige analyse van mogelijke voordelen toeliet maar slechts een kwalitatieve indicatie.

Gezien de bovengenoemde doelen en keuzes was het niet nodig een compleet systeem uit te ontwikkelen met een toegesneden gebruikersinterface,

waarmee modellen en specificaties kunnen worden gecreëerd en simulaties van gebruiksprocessen kunnen worden uitgevoerd. In plaats daarvan hebben mijn inspanningen zich geconcentreerd op het realiseren en combineren van de volgende deelfuncties: (i) modelleren en simuleren met geharmoniseerde kennisbronnen (d.w.z. het toevoegen van transitierelaties aan nucleusgebaseerd modelleren en simuleren), en (ii) simulatieaansturing met scenariobundels.

Modelleren en simuleren met geharmoniseerde kennisbronnen gaat uit van nucleusgebaseerd modelleren en simuleren, waarvoor nog geen toegesneden softwarecomponenten bestonden. In plaats van softwarecomponenten te ontwikkelen voor alle deelfuncties heb ik ervoor gekozen om uit te gaan van substituuft-software die soortgelijke functionaliteit biedt. Om simuleerbare nucleusgebaseerde modellen te emuleren werd het dynamicapakket msc Adams gebruikt. Transitierelaties zijn als toestandstransities gespecificeerd en gemodelleerd met Matlab Simulink door gebruik te maken van de statechart-notatie. Simulink is ook gebruikt om de transities tussen situaties te doorlopen en zo de Adams-simulatie aan te sturen.

Verificatie. Om de theorie en de implicaties daarvan in toepassingen op juistheid en geloofwaardigheid te toetsen heb ik de correctheid onderzocht door de consequenties van de theorie te overwegen. De beweringen in elke hypothese werden geanalyseerd door te na te gaan onder welke condities zij hun geldigheid zouden verliezen. Als resultaat konden op overtuigende wijze categorieën van toepassingsgebieden worden benoemd waarvoor de aannames in de hypothesen niet gesteund worden door de theorie.

De beperkingen konden worden ondergebracht in twee hoofdcategorieën: (i) denkbare behoeften van ontwerpers waarvoor de theorie geen oplossing biedt en (ii) groepen van productsoorten en gebruiksvormen waarvoor de theorie niet geldt. Een beperking van de eerste soort is bijvoorbeeld dat de ontwerper geen modellen kan veranderen terwijl de simulatie loopt omdat bij zulke veranderingen de veronderstelde relaties tussen opeenvolgende interacties vervallen. Een beperking van de tweede soort is bijvoorbeeld dat het niet mogelijk is om gebruiksprocessen te simuleren en aan te sturen die afhangen van leereffecten en opgedane ervaring, omdat toestandstransities beslissingsprocessen die meerdere gebruiksprocessen overstijgen niet kunnen beschrijven.

De inventarisatie van beperkingen was niet uitputtend. Verder systematisch onderzoek zal andere nog niet in kaart gebrachte beperkingen aan het licht moeten brengen.

Validatie. De voorgestelde aanpak en de pilotimplementatie zijn op de volgende twee punten gevalideerd: (i) de nieuwe functionaliteiten die de aanpak volgens de hypothesen biedt en (ii) het praktische nut van het geïmplementeerde systeem op basis van technische beoordeelbare aspecten van gebruiksgemak.

Daartoe heb ik volgens de voorgestelde aanpak voorbeeldproducten gespecificeerd en gemodelleerd, en vervolgens gesimuleerd met het pilotsysteem. Deze producten voldeden aan de kenmerken (i) dat het gebruik ervan verscheidene opeenvolgende interacties omvat en (ii) dat er verschillende interactievolgordes mogelijk waren. Om de multifysica-ondersteuning die nucleusgebaseerd model-

leren en simuleren biedt te demonstreren was een aanvullende eis dat (iii) het gebruik afhangt van een combinatie van fysische verschijnselen die niet met bestaande simulatiemethoden kunnen worden bestudeerd.

Het als laatste en meest uitgebreid behandelde gebruiksvoorbeeld betreft een snackdispenser. Dit voorbeeld omvatte (i) een reeks van handelingen bestaande uit het reiken naar een punt, het indrukken en loslaten van een knop, het beetpakken van een object en dit vervolgens optillen en wegnemen, en (ii) variatiemogelijkheden in de interactievolgorde, die konden worden gerealiseerd door te spelen met een instelling voor de tijd gedurende welke de menselijke gebruiker aarzelt te reageren. Bovendien (iii) was in de simulatie van het gebruiksproces dynamica van starre lichamen gecombineerd met grote vervormingen (in de vingers, tijdens beetpakken).

Om de aanpak en de pilotimplementatie te iken is een referentieaanpak gedefinieerd die beschrijft hoe de nieuw geboden functionaliteit zo goed mogelijk kan worden benaderd door uitsluitend gebruik te maken van conventionele dynamica-simulatiesoftware.

Met betrekking tot evaluatie van de vernieuwende functionaliteiten kon het demonstratieve voorbeeld aantonen dat het voorgestelde logische aansturingsprincipe het mogelijk maakte om verschillende gebruiksscenario's te beramen en om het verloop van de gesimuleerde scenario's te volgen. In tegenstelling tot de referentieaanpak met conventionele simulatiesoftware kan het aansturingsprincipe (i) vanuit een enkele scenariobundel meerdere simulaties met verschillende aaneenschakelingen van interacties voortbrengen en (ii) ook bij bescheiden veranderingen in de productgeometrie en/of karakteristieken van de menselijke gebruiker worden toegepast, zonder dat aanpassing van de scenariobundel nodig was.

De bruikbaarheid van het systeem in termen van gebruiksgemak werd geobjectiveerd door twee kwantitatieve indicatoren te definiëren, een voor de benodigde inspanningen om simulaties voor te bereiden en een voor de simulatietijd.

De *preparation efforts indicator* (PEI) drukt de voorbereidingsinspanningen uit op basis van het voor de ontwerper benodigde aantal handelingen, als een verhouding tussen wat nodig is voor de scenariobundel-aanpak en voor de referentieaanpak. Met deze indicator kon worden aangetoond dat er op voorbereidingstijd wordt bespaard als één scenario met verscheidene (ten minste vijf) variaties van het product- of gebruikersmodel wordt gesimuleerd. De tijdsbesparing neemt toe met het aantal transities tussen interacties die in het scenario zijn vastgelegd.

De simulatietijd-indicator (STI) drukt de totale benodigde procestijd uit die de computer nodig heeft als een verhouding tussen wat nodig is voor de scenariobundel-aanpak en voor de referentieaanpak. Deze blijkt een functie te zijn van (i) de procestijd die nodig is om de fysica van een gegeven gebruiksproces te simuleren, (ii) de procestijd nodig voor verwerking van logische beheers- en regelinstructies in datzelfde proces, (iii) het aantal transities tussen interacties in het proces en (iv) het aantal bewegende delen in het mens-productsysteem. Er kon worden aangetoond dat als het te simuleren systeem twee of meer bewe-

gende delen bevat, simulaties op basis van scenariobundels minder rekentijd vergen. De voorsprong neemt toe als het aantal onderdelen verder toeneemt en als het aantal transities toeneemt. Voor het hierboven beschreven kenmerkende gebruiksproces van het voorbeeldproduct bleek de waarde van de STI ongeveer gelijk te zijn aan 10, d.w.z. dat de aanpak met scenariobundels de rekentijd met ongeveer 90% terugbrengt ten opzichte van de referentieaanpak.

Discussie. Terugblikkend op de gebruikte onderzoeksmethoden bleek de beslissing om commercieel verkrijgbare substituut-software te gebruiken de meest kritieke invloedsfactor op de uitvoering van het onderzoek. De belangrijkste positieve consequentie is geweest dat binnen de beschikbare tijd voor een promotieproject uitgevoerd door één persoon een pilotsysteem kon worden gerealiseerd dat voldoende kon overtuigen en dat een zinvolle validatie toeliet.

Deze keuze had twee belangrijke negatieve consequenties. De eerste was dat de fysicasimulatie zeer gevoelig bleek te zijn voor kleine veranderingen in modellen en logische transitiecondities. Dit werd veroorzaakt door (i) het drastisch gesimplificeerde mensmodel, dat moest worden ingezet bij afwezigheid van natuurgetrouwe modellen van menselijke motoriek en (ii) de noodgrepen die moesten worden toegepast om modellen in Adams op te bouwen uit partikels, en die nodig waren om met deze simulatiesoftware voor starre lichamen toch flexibiliteit te kunnen modelleren. Vanwege deze gevoeligheid moesten door herhaald uitproberen voortdurend nieuwe stabiele waarden voor parameters in de modellen en specificaties worden gezocht. Bijgevolg ontbrak de tijd om andere voorbeeldproducten met hun gebruiksprocessen uit te werken. Door meer casussen met verschillende voorbeeldproducten te bestuderen (bij voorkeur in echte ontwerpprocessen) zou er aan voordelen van het simuleren met scenariobundels statistische significantie kunnen worden toegekend. Er is echter onzekerheid over het vereiste significantieniveau en hoe dit kan worden bereikt.

De tweede negatieve consequentie was dat een op maat gemaakte gebruiksinterface ontbrak waarmee de aanpak kon worden uitgetoetst in de ontwerppraktijk.

Conclusies. Om gebruiksprocessen te simuleren en daarbij reeksen van interacties te ordenen en met elkaar te verbinden kan het beginsel van harmonisatie van kennisbronnen worden toegepast. Dit biedt ontwerpers de mogelijkheid om transitierelaties te gebruiken bij het specificeren van menselijke interacties in scenariobundels, d.w.z. zelfbedachte geordende stelsels van scenario's, die voorbouwen op een vereenvoudigd redeneermodel over mens-productinteractie. Met bestaande software is een pilotimplementatie gerealiseerd waarmee kon worden aangetoond dat deze aanpak met succes kon worden toegepast om (i) gecombineerde opeenvolgingen van primaire fysieke interacties (scenario's) te simuleren, (ii) verscheidene scenario's te simuleren die in één bundel zijn vastgelegd en (iii) met variaties van het product (en/of de gebruiker of de omgeving) simulaties uit te voeren. De systemen die momenteel op de markt zijn bieden deze mogelijkheden niet. Als we kijken naar de voorbereidende inspanningen en de rekentijd benodigd voor simulaties blijkt de voorgestelde aanpak voldoende voordeel te bieden indien complexe gebruiksprocessen met meerdere transities tussen inter-

acties onderzocht worden. Hier moet wel bij gezegd worden dat de aanpak niet bedoeld is voor simulaties waarin zulke transities geen rol spelen en waarvoor conventionele methoden reeds voldoen.

Om het promotieonderzoek binnen redelijke tijd af te ronden moesten enkele belangrijke kwesties worden doorgeschoven naar vervolgonderzoek en -ontwikkeling. Daarin verdienen de volgende punten aandacht: (i) realisatie van een compleet systeem dat met ontwerpers kan worden getest en voorzien is van een toegesneden gebruikersinterface, (ii) het inbouwen van meer detailkennis over menselijke motoriek en (iii) onderzoek naar mogelijkheden om de inspanningen die nodig zijn voor het opzetten van simulaties te reduceren door stappen te automatiseren en voorgedefinieerde sjabloonmodellen en -specificaties beschikbaar te maken.

REFERENCES

- [1] Pew, R.W., and Mavor, A.S. (2007), Human-system integration in the system development process: a new look, The National Academic Press, Washington, DC.
- [2] Carruth, D.W., and Duffy, V.G. (2005), "Towards integrating cognitive models and digital human models", Proceedings of the 11th International Conference on Human-Computer Interaction, Las Vegas.
- [3] Carroll, J.M., Mack, R.L., Robertson, S.P., and Rosson, M.B. (1994), "Binding objects to scenarios of use", International Journal Human-Computer Studies, Vol. 41, pp. 243-279.
- [4] Welker, K., Sanders, E.B.N., and Couch, J.S. (1997), "Design scenarios - to understand the user", Innovation (3), pp. 24-27.
- [5] Landay, J.A., and Myers, B.A. (1996), "Sketching storyboards to illustrate interface behaviors", Proceedings of the Human Factors in Computing Systems: CHI '96, Vancouver, pp. 193-194.
- [6] Kankainen, A. (2003), "UCPCD: User-Centered Product Concept Design", Proceedings of the ACM Conference on Designing for User Experiences, San Francisco, pp. 1-13.
- [7] Cremer, J., Kearney, J., and Ko, H. (1996), "Simulation and scenario support for virtual environments", Computers & Graphics, Vol. 20 (2), pp. 199-206.
- [8] Hooper, J., and Hsia, P. (1982), "Scenario-based prototyping for requirements identification", Software Engineering Notes, Vol. 7 (5), pp. 88-93.
- [9] Korn, G.A., and Wait, J.V. (1978), Digital continuous system simulation, Prentice-Hall, Englewood Cliffs.
- [10] Law, A.M., and Kelton, W.D. (1991), Simulation modeling & analysis, 2nd ed., McGraw-Hill, New York.
- [11] Mutha, P.K., and Sainburg, R.L. (2007), "Control of velocity and position in single joint movements", Human Movement Science, Vol. 26, pp. 808-823.
- [12] Jiang, J., Shen, Y., and Neilson, P.D. (2002), "A simulation study of the degrees of freedom of movement in reaching and grasping", Human Movement Science, Vol. 21, pp. 881-904.
- [13] Zeigler, B.P., Praehofer, H., and Kim, T.G. (2000), Theory of modeling and simulation - integrating discrete event and continuous complex dynamic systems, second edition, Academic press, San Diego.
- [14] Anderson, J.R., Bothell, D., Byrne, M.D., and Lebiere, C. (2004), "An integrated theory of the mind", Psychological Review, Vol. 111 (4), pp. 1036-1060.
- [15] Webb, J.N. (2007), Game theory - decisions, interaction and evolution, Springer, London.
- [16] Boess, S.U. (2009), "Experiencing product use in product design", Proceedings of the ICED, Vol. 9, pp. 311-321.
- [17] Varga, E. (2008), "Using Hand Motions in Conceptual Shape Design: Theories, Methods and Tools", Product Engineering: Tools and Methods Based on Virtual Reality, pp. 367-382.
- [18] Verlinden, J., Horváth, I., and Nam, T. (2009), "Recording augmented reality experiences to capture design reviews", International Journal on Interactive Design and Manufacturing, Vol. 3 (3), pp. 189-200.

- [19] Opiyo, E.Z., Horváth, I., and Rusák, Z. (2009), "Strategies for model simplification and data reduction in holographic virtual prototyping and product visualization through application dependent model pre-processing", Proceedings of the ASME-CIE, San Diego.
- [20] Moes, C.C.M. (2004), Advanced human body modelling to support designing products for physical interaction, PhD thesis, Delft University of Technology, Delft.
- [21] Zhang, B. (2005), Using artificial neural networks for prediction of human body postures based on 3D landmarks, PhD thesis, Delft University of Technology, Delft.
- [22] Rusák, Z., Antonya, C., Horváth, I., and Talaba, D. (2009), "Comparing kinematic and dynamic hand models for interactive grasping simulation", Proceedings of the ASME-CIE, San Diego.
- [23] Rusák, Z. (2003), Vague discrete interval modeling for product conceptualization in collaborative virtual environments, PhD thesis, Millpress, Rotterdam.
- [24] Rusák, Z., Horváth, I., and Van der Vegte, W.F. (2004), "First steps towards an all embracing relations-based modelling", Proceedings of the ASME-DETC, Salt Lake City, UT.
- [25] Horváth, I. (2008), "Differences between 'research in design context' and 'design inclusive research' in the domain of industrial design engineering", Journal of Design Research, Vol. 7 (1), pp. 61-83.
- [26] Van der Vegte, W.F. (2006), "A survey of artifact-simulation approaches from the perspective of application to use-processes of consumer durables", Proceedings of the TMCE, Horváth, I., Rusák, Z. (Eds.), Vol. 2, Ljubljana, Slovenia, pp. 617-632.
- [27] Van der Vegte, W.F., and Horváth, I. (2006), "Including human behavior in product simulations for the investigation of use processes in conceptual design: A survey", Proceedings of the ASME-CIE, Philadelphia, Pennsylvania.
- [28] Horváth, I., and Van der Vegte, W.F. (2003), "Nucleus-based product conceptualization - Principles and formalization", Proceedings of the ICED, Stockholm, Sweden.
- [29] Van der Vegte, W.F., and Horváth, I. (2003), "Nucleus-based product conceptualization - Application in Designing for Use", Proceedings of the ICED, Stockholm, Sweden.
- [30] Van der Vegte, W.F., and Horváth, I. (2003), "Use-driven product conceptualization based on nucleus modeling and simulation with scenarios", Proceedings of the 15th European Simulation Symposium, Delft, The Netherlands.
- [31] Van der Vegte, W.F., and Rusák, Z. (2007), "Hybrid simulation of use processes with scenario structures and resource-integrated models", Proceedings of the ASME-CIE, Las Vegas, Nevada.
- [32] Van der Vegte, W.F., Horváth, I., and Rusák, Z. (2008), "Simulating the use of products: applying the nucleus paradigm to resource-integrated virtual interaction models", Proceedings of the TMCE, Horváth, I., Rusák, Z. (Eds.), Vol. 1, Izmir, pp. 591-605.
- [33] Van der Vegte, W.F., and Rusák, Z. (2008), "Controlling simulations of human-artifact interaction with scenario bundles", Proceedings of the EDIPROD'08, Jurata, pp. 197-209.
- [34] Van der Vegte, W.F., Pulles, J.P.W., and Vergeest, J.S.M. (2001), "Towards computer-supported inclusion and integration of life cycle processes in product conceptualization based on the process tree", Automation in Construction, Vol. 10/6, pp. 731-740.

- [35] Van der Vegte, W.F., Vergeest, J.S.M., and Horváth, I. (2001), "Towards a unified description of product related processes", *Transactions of the SDPS - Journal of Integrated Design & Process Science*, Vol. 5 (2), pp. 53-63.
- [36] Van der Vegte, W.F., and Horváth, I. (2002), "Consideration and modeling of use processes in computer-aided conceptual design", *Transactions of the SDPS - Journal of Integrated Design & Process Science*, Vol. 6 (2), pp. 25-59.
- [37] Van der Vegte, W.F., Kitamura, Y., Koji, Y., and Mizoguchi, R. (2004), "Coping with unintended behavior of users and products: ontological modelling of product functionality and use", *Proceedings of the ASME-CIE*, Salt Lake City, Utah.
- [38] Palm, W.J.I. (2005), *System dynamics*, McGraw Hill, New York.
- [39] Knepell, P.L., and Aragno, D.C. (1993), *Simulation validation - a confidence assessment methodology*, IEEE Computer Society Press, Los Alamitos, CA.
- [40] Chen, X., Salem, M., Das, T., and Chen, X. (2008), "Real time software-in-the-loop simulation for control performance validation", *Simulation*, Vol. 84 (8-9), pp. 457-471.
- [41] Voet, D., Voet, J.G., and Pratt, C.W. (2006), *Fundamentals of biochemistry*, Wiley, New York.
- [42] Anonymous (2004), "Modeling Muscle Fatigue", in: *End-of-year technical report for project digital human modeling and virtual reality for future combat systems*, Abdel-Malek, K. (Ed.), University of Iowa, Iowa City.
- [43] McLeod, I.S. (2004), "Human factors verification and validation", in: *Human factors for engineers*, Sandom, C., Harvey, R.S. (Eds.), The Institution of Electrical Engineers, London.
- [44] Strauss, R.H. (1979), *Sports medicine and physiology*, Saunders, Philadelphia.
- [45] Hui, Y.H. (2006), *Food biochemistry and food processing*, Blackwell Publishing, Ames.
- [46] Day, R.H. (1969), *Human Perception*, Wiley, Sydney.
- [47] Robinson-Riegler, G., and Robinson-Riegler, B. (2004), *Cognitive psychology: applying the science of the mind*, Pearson, Boston.
- [48] Rasmussen, J. (1986), *Information processing and human-machine interaction*, North-Holland, New York.
- [49] Oatis, C.A. (2004), *Kinesiology - the mechanics & pathomechanics of human movement*, Lippincott Williams & Wilkins, Philadelphia.
- [50] Silver, F.H. (1987), *Biological materials: structure, mechanical properties, and modeling of soft tissues*, New York University Press, New York.
- [51] Bryant, C.R., Kurfman, M.A., Stone, R.B., and McAdams, D.A. (2001), "Creating equation handbooks to model design performance parameters", *Proceedings of the ICED*, Glasgow, pp. 501-508.
- [52] Gelsey, A. (1991), "From CAD/CAM to simulation: automatic model generation for mechanical devices", in: *Knowledge-based simulation - methodology and application*, Fishwick, P.A., Modjeski, R.B. (Eds.), Springer, New York, pp. 108-132.
- [53] Roth, K. (1982), *Konstruieren mit Konstruktionskatalogen*, Springer, Berlin.
- [54] Baldwin, D., Göktas, Ü., Hereman, W., Hong, L., Martino, R.S., and Miller, J.C. (2004), "Symbolic computation of exact solutions expressible in hyperbolic and elliptic functions for nonlinear PDEs", *Journal of Symbolic Computation*, Vol. 37, pp. 669-705.
- [55] Shigley, J.E., and Mischke, C.R. (1989), *Mechanical Engineering Design*, McGraw-Hill, New York.

- [56] Meriam, J.L., and Kraige, L.G. (2003), *Engineering mechanics*, Vol. 1 and 2, Wiley, Hoboken.
- [57] Jeon, W.H., Baek, S.J., and Kim, C.J. (2003), "Analysis of the aeroacoustic characteristics of the centrifugal fan in a vacuum cleaner", *Journal of Sound and Vibration*, Vol. 268 (5), pp. 1025-1035.
- [58] Huston, R.L., and Passarello, C.E. (1982), "The mechanics of human body motion (due to impulsive forces during occupational manoeuvres and jumping)", in: *Human body dynamics - impact, occupational and athletic aspects*, Ghista, D.N. (Ed.), Clarendon Press, Oxford.
- [59] Hughes, R.E., and An, K.N. (1999), "Biomechanical models of the hand, wrist and elbow in ergonomics", in: *Biomechanics in ergonomics*, Kumar, S. (Ed.), Taylor & Francis, London, pp. 179-198.
- [60] Griffin, M.J. (1990), *Human vibration*, Academic Press, London.
- [61] Job, H.M., Keating, D., Evans, A.L., and Parks, S. (1999), "Three-dimensional electromagnetic model of the human eye: advances towards the optimisation of electroretinographic signal detection", *Medical & Biological Engineering & Computing*, Vol. 37, pp. 710-719.
- [62] Mochnacki, B., and Majchrzak, E. (2003), "Sensitivity of skin tissue on the activity of external heat sources", *Computer Modeling in Engineering and Sciences*, Vol. 4, pp. 431-438.
- [63] Panescu, D., Webster, J.G., and Stratbucker, R.A. (1994), "A nonlinear electrical-thermal model of the skin", *IEEE Transactions on Biomedical Engineering*, Vol. 41 (7), pp. 672-680.
- [64] Therrien, R.G., and Bourassa, P.A. (1982), "Mechanics application to sports equipment: protective helmets, hockey sticks, and jogging shoes", in: *Human body dynamics - impact, occupational and athletic aspects*, Ghista, D.N. (Ed.), Clarendon Press, Oxford.
- [65] Joines, J.A., and Roberts, S.D. (1998), "Fundamentals of object-oriented simulation", *Proceedings of the 1998 winter Simulation Conference*, pp. 141-149.
- [66] Pllana, S., and Fahringer, T. (2002), "UML-based modeling of performance oriented parallel and distributed applications", *Proceedings of the Winter Simulation Conference*, pp. 497-505.
- [67] Pop, A., Savga, I., Aßmann, U., and Fritzson, P. (2005), "Composition of XML dialects: A Modelica XML case study", *Electronic Notes in Theoretical Computer Science*, Vol. 11, pp. 137-152.
- [68] Sinha, R., V.C. Liang, Paredis, C.J.J., Liang, V.-C., and Khosla, P.K. (2001), "Modeling and simulation methods for design of engineering systems", *Journal of Computing and Information Science in Engineering*, Vol. 1 (1), pp. 84-91.
- [69] Riley, K.F., Hobson, M.P., and Bence, S.J. (1997), *Mathematical methods for physics and engineering*, Cambridge Press, Cambridge.
- [70] Baraff, D. (1994), "Fast contact force computation for nonpenetrating rigid bodies", *Proceedings of the SIGGRAPH*, Orlando.
- [71] Hummel, A., and Girod, B. (1997), "Fast dynamic simulation of flexible and rigid bodies with kinematic constraints", *Proceedings of the ACM VRST 1997*.
- [72] Bobrow, D.G. (1984), "Qualitative reasoning about physical systems: an introduction", *Artificial Intelligence*, Vol. 24, pp. 1-5.
- [73] Forbus, K.D. (1984), "Qualitative Process Theory", *Artificial Intelligence*, Vol. 24, pp. 85-168.
- [74] Kuipers, B.J. (2001), "Qualitative simulation", in: *Encyclopedia of Physical Science and Technology*, 3rd edition, Meyers, A. (Ed.), Academic Press, New York, pp. 287-300.

- [75] Cellier, F.E., and Roddier, N. (1991), "Qualitative State Spaces: A Formalization of the Naïve Physics Approach to Knowledge-based Reasoning", *Proceedings of the AI, Simulation and Planning in High Autonomy Systems*, Cocoa Beach, pp. 40-49.
- [76] Bouwer, A., and Bredeweg, B. (2008), "Graphical means for inspecting qualitative models of system behaviour", *Instructional Science* (December), pp. 1-36.
- [77] Kramer, G.A., Barrow, H.G., and Agre, P.E. (1993), "Qualitative kinematics", *United States Patent No. 5,253,189*.
- [78] Bozzo, L.M., Barbat, A., and Torres, L. (1998), "Application of qualitative reasoning in engineering", *Applied Artificial Intelligence*, Vol. 12, pp. 29-48.
- [79] Šuc, D., and Bratko, I. (2000), "Qualitative trees applied to bicycle riding", *Linköping Electronic Transactions on Artificial Intelligence*, Vol. 4 (B), pp. 125-140.
- [80] Lucas, P. (Ed.), (2003), *Model-based and Qualitative Reasoning in Biomedicine: Working notes of the workshop held during The 9th European Conference on Artificial Intelligence in Medicine, AIME'03*, Protaras.
- [81] De Kleer, J., and Brown, J.S. (1984), "A qualitative physics based on confluences", *Artificial Intelligence*, Vol. 24, pp. 87-83.
- [82] Xia, S., Linkens, D.A., and Bennett, S. (1993), "Automatic modelling and analysis of dynamic physical systems using qualitative reasoning and bond graphs", *Intelligent Systems Engineering*, Vol. 3 (2), pp. 201-212.
- [83] Bouwer, A., and Bredeweg, B. (2001), "VisiGarp: graphical representation of qualitative simulation models", *Proceedings of the International Qualitative Reasoning Workshop*.
- [84] Clancy, D.J., and Kuipers, B.J. (1994), "Model decomposition and simulation", *Proceedings of the International Qualitative Reasoning Workshop*.
- [85] Paynter, H.M. (1961), *Analysis and design of engineering systems*, The MIT Press, Cambridge, MA.
- [86] Beukeboom, J.A.J.J., Van Dixhoorn, J.J., and Meerman, J.W. (1985), "Simulation of mixed bond graphs and block diagrams on personal computers using TUTSIM", *Journal of the Franklin Institute*, Vol. 319, pp. 257-267.
- [87] Finger, S., Chan, X., Lan, R., and Cahn, B. (2001), "Creating virtual prototypes - integrating design and simulation", *Proceedings of the ICED*, Glasgow, pp. 485-492.
- [88] Fishwick, P.A. (1995), *Simulation model design and execution: building digital worlds*, Prentice-Hall, Englewood Cliffs.
- [89] Karaynakis, N.M. (1995), *Advanced system modelling and simulation with block diagram languages*, CRC Press, Boca Raton.
- [90] Brück, D., Elmqvist, H., Olsson, H., and Matteson, S.E. (2002), "Dymola for multi-engineering modeling and simulation", *Proceedings of the 2nd International Modelica Conference*, Vol. 55, Oberpfaffenhofen, pp. 1-8.
- [91] Cellier, F.E., and Nebot, À. (2005), "The Modelica bond graph library", *Proceedings of the 4th International Modelica Conference*, Hamburg, pp. 57-65.
- [92] Montbrun-Di Filippo, J., Delgado, M., Brie, C., and Paynter, H.M. (1991), "A survey of bond graphs : Theory, applications and programs", *Journal of the Franklin Institute*, Vol. 328 (5-6), pp. 565-606.
- [93] Swider, J., and Wszolek, G. (2004), "Analysis of complex mechanical systems based on the block diagrams and the matrix hybrid graph method", *Journal of Materials Processing technology*, Vol. 157-158, pp. 250-255.

- [94] Remmerswaal, J.A.M., and Pacejka, H. (1985), "A bond graph computer model to simulate vacuum cleaner dynamics for design purposes", *Journal of the Franklin Institute*, Vol. 319, pp. 83-92.
- [95] Margolis, D.L. (1975), "An introduction to bond-graph techniques for biomedical system modeling", *Simulation*, Vol. 44, pp. 22-26.
- [96] Wojcik, L.A. (2003), "Modeling of musculoskeletal structure and function using a modular bond graph approach", *Journal of the Franklin Institute*, Vol. 340, pp. 63-76.
- [97] Pop, C., Khajepour, A., Huisson, J.P., and Patla, A.E. (1999), "Application of Bond graphs to Human Locomotion Modeling", *Proceedings of the HKK Conference and Symposium*, Waterloo, pp. 85-90.
- [98] Hubbard, M. (1980), "Dynamics of the pole vault", *Journal of Biomechanics*, Vol. 13, pp. 965-976.
- [99] Ferretti, G., Magnani, G.A., and Rocco, P. (2004), "Virtual prototyping of mechatronic systems", *Annual Reviews in Control*, Vol. 28, pp. 193-206.
- [100] Evers, W.J.E., Besselink, I.J.M., Van der Knaap, A.C.M., and Nijmeijer, H. (2009), "Development and validation of a modular simulation model for commercial vehicles", *International Journal of Heavy Vehicle Systems*, Vol. 16 (1-2), pp. 132-153.
- [101] Thoma, J., and Halin, H.J. (1999), "Bond graphs and practical simulation", *Simulation Practice and Theory*, Vol. 7, pp. 401-417.
- [102] Yen, C., and Masada, G.Y. (1991), "Model of a hyperelastic thin plate using the extended bond graph method", *Journal of the Franklin Institute*, Vol. 328 (5-6), pp. 765-780.
- [103] McCarthy, J.M. (2000), *Geometric design of linkages*, Springer-Verlag, New York.
- [104] Horváth, I., Vergeest, J.S.M., and Kuczogi, G. (1998), "Development and application of design concept ontologies for contextual conceptualization", *Proceedings of the ASME-CIE 1998*.
- [105] Horváth, I., Thernes, V., and Bagoly, Z. (1995), "Conceptual design with functionally and morphologically parameterized feature objects", *Proceedings of the ASME-CIE*.
- [106] Krovi, V., Ananthasuresh, G.K., and Kumar, V. (2002), "Kinematic and kinetostatic synthesis of planar coupled serial chain mechanisms", *ASME Journal of Mechanical Design*, Vol. 124, pp. 301-312.
- [107] Horváth, I., Zhang, B., Moes, C.C.M., and Molenbroek, J.M. (2005), "Status of digital human body modeling from the aspect of information inclusion", *Proceedings of the ASME-CIE, Long Beach CA*.
- [108] Jung, E.S., and Choe, J. (1996), "Human reach posture prediction based on psychophysical discomfort", *International Journal of Industrial Ergonomics*, Vol. 18, pp. 173-179.
- [109] Lee, K. (1999), *Principles of CAD / CAM / CAE systems*, Addison Wesley, Reading MA.
- [110] Gonzalez, M., Gonzalez, F., Luaces, A., and Cuadrado, J. (2007), "Interoperability and neutral data formats in multibody system simulation", *Multibody System Dynamics*, Vol. 18 (1), pp. 59-72.
- [111] Schiehlen, W. (1997), "Multibody system dynamics: roots and perspectives", *Multibody System Dynamics*, Vol. 1 (2), pp. 149-188.
- [112] Wang, S.L. (2001), "Motion simulation with Working Model 2D and MSC.visualNastran 4D", *Journal of Computing and Information Science in Engineering*, Vol. 1 (6), pp. 193-196.
- [113] Gissinger, G.L., and Kortum, W. (1997), "Simulation of vehicle system dynamics, state of the art and ongoing developments", *Transportation Systems 1997*, Vols 1-3, pp. 201-208.

- [114] Limebeer, S.J.N., and Sharp, R.S. (2007), "Bicycles, motorcycles, and models (vol 26, pg 34, 2006)", *IEEE Control Systems Magazine*, Vol. 27 (3), pp. 118-118.
- [115] Nho, G.H., Yoo, W.S., Chung, B.S., Kan, D.W., and Lyu, J.C. (2006), "Matching of multibody dynamic simulation and experiment of a drum-type washing machine", *Proceedings of the Third Asian Conference on Multibody Dynamics*, Tokyo, pp. 662-667.
- [116] Sanders, M.S., and McCormick, E.J. (1992), *Human factors in Engineering and Design*, McGraw Hill, Singapore.
- [117] Rasmussen, J., Damsgaard, M., Surma, E., Christensen, S.T., De Zee, M., and Vonderak, V. (2003), "Anybody-a software system for ergonomic optimization", *Proceedings of the Fifth World Congress on Structural and Multidisciplinary Optimization*, Venice.
- [118] Kenny, I.C., Wallace, E.S., Otto, S.R., and Brown, D. (2005), "Validation of a full-body computer simulation model for the golf drive", *Proceedings of the Annual Conference of the Exercise and Sport Sciences Association of Ireland*, Munich.
- [119] Feyen, R., Liu, L., Chaffin, D., Jimmerson, G., and Joseph, B. (2000), "Computer-aided ergonomics: a case study of incorporating ergonomics analyses into workplace design", *Applied Ergonomics*, Vol. 31, pp. 291-300.
- [120] Bern, M., and Plassmann, P. (2000), "Mesh Generation", in: *Handbook of Computational Geometry*, Sack, J., Urrutia, J. (Eds.), North Holland, New York.
- [121] Zeid, I. (1991), *CAD/CAM theory and practice*, McGraw-Hill, New York.
- [122] Zienkiewicz, O.C., and Hollister, G.S. (1965), *Stress analysis, recent developments in numerical and experimental methods*, Wiley, New York.
- [123] Zienkiewicz, O.C., and Taylor, R.L. (2000), *The finite element method*, Butterworth-Heinemann, Oxford.
- [124] Tsuchiya, T., Kagawa, Y., Doi, M.a.T., and T (2003), "Finite element simulation of non-linear acoustic generation in a horn loudspeaker", *Journal of Sound and Vibration*, Vol. 266 (5), pp. 993-1008.
- [125] Fikri, R., Barchiesi, D., H'Dhili, F., Bachelot, R., Vial, A., and Royer, P. (2003), "Modeling recent experiments of apertureless near-field optical microscopy using 2D finite element method", *Optics Communications*, Vol. 221, pp. 13-22.
- [126] Kita, E., and Kamiya, N. (2001), "Error estimation and adaptive mesh refinement in boundary element method, an overview", *Engineering Analysis with Boundary Elements*, Vol. 25, pp. 479-495.
- [127] Puig-Pey, J., and Brebbia, C.A. (1987), "Introduction to CAE systems", in: *Computer Aided engineering Systems Handbook*, Puig-Pey, J., Brebbia, C.A. (Eds.), Vol. I, Springer, Heidelberg, pp. 1-10.
- [128] Aubel, A. (2004), "Body deformations", in: *Handbook of virtual humans*, Magnenat-Thalmann, N., Thalmann, D. (Eds.), Wiley, Hoboken, pp. 140-160.
- [129] Lewis, J.P., Cordner, M., and Fong, N. (2000), "Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation", *Proceedings of the SIGGRAPH*, pp. 165-172.
- [130] Versteeg, H.K., and Malalasekera, W. (1995), *An introduction to computational fluid dynamics; the finite volume method*, Longman Scientific and Technical, Harlow.
- [131] Mackerle, J. (2000), "Parallel finite element and boundary element analysis: theory and applications - a bibliography (1997-1999)", *Finite Elements in Analysis and Design*, Vol. 35, pp. 283-296.

- [132] Komzsik, L., and Stanton, E. (2000), "Trends in analysis and optimization of products", *Proceedings of the TMCE, Delft*, pp. 19-27.
- [133] Bailey, C., Taylor, G.A., Cross, M., and Chow, P. (1999), "Discretisation procedures for multi-physics phenomena", *Journal of Computational and Applied Mathematics*, Vol. 103 (1), pp. 3-17.
- [134] Cross, M., Croft, T.N., Slone, A.K., Williams, A.J., Christakis, N., Patel, M.K., Bailey, C., and Pericleous, K. (2007), "Computational modelling of multi-physics and multi-scale processes in parallel", *International Journal for Computational Methods in Engineering Science and Mechanics*, Vol. 8, pp. 1-12.
- [135] Cross, M., and Slone, A.K. (2005), "FENet - multi-physics analysis (MPA) theme: a review of commercial MPA capability in 2005", *Proceedings of the FENet project review meeting, St. Julians, Malta*, pp. 119-130.
- [136] Zhu, C., Zhu, L., Liu, Y., and Cai, G. (2007), "Dynamics Modeling and Co-simulation of Rigid-flexible Coupling System of 3-TPT Parallel Robot", *Proceedings of the IEEE International Conference on Automation and Logistics, Jinan*, pp. 2627-2632.
- [137] Friswell, M.I., Smart, M.G., and Hamblyn, S.M. (1996), "The validation and updating of dynamic models of golf clubs", *Proceedings of the first International Conference on the Engineering of Sport, Sheffield*.
- [138] Middendorf, W.H. (1990), *Design of devices and systems*, Marcel Dekker, New York.
- [139] Hanna, R.K. (1996), "Going faster, higher and longer in sport with CFD", *Proceedings of the first International Conference on the Engineering of Sport, Sheffield*.
- [140] Les, C.M., Keyak, J.H., Stover, S.M., and Taylor, K.T. (1997), "Development and validation of a series of three-dimensional finite element models of the equine metacarpus", *Journal of Biomechanics*, Vol. 7, pp. 737-742.
- [141] Chow, W.W., and Odell, E.I. (1977), "Deformations and stresses in soft body tissues of a sitting person", *Journal of Biomechanical Engineering*, May 1978, Vol. 100, pp. 79-87.
- [142] Cheung, J.T.M., Zhang, M., Leung, A.K.L., and Fan, Y.B. (2005), "Three-dimensional finite element analysis of the foot during standing - a material sensitivity study", *Journal of Biomechanics*, Vol. 38, pp. 1045-1054.
- [143] Bandak, F.A., Tannous, R.E., and Toridis, T. (2001), "On the development of an osseo-ligamentous finite element model of the human ankle joint", *Int. Journal of Solids and Structures*, Vol. 38, pp. 1681-1697.
- [144] Koch, R.M., Gross, M.H., and Bosshard, A.A. (1998), "Emotion editing using finite elements", *Eurographics*, Vol. 17 (3), pp. 295-302.
- [145] Hrúz, B., and Zhou, M.C. (2007), *Modeling and control of discrete-event dynamic systems*, Springer, London.
- [146] Sun, Q., Gan, R.Z., Chang, K.H., and Dormer, K.J. (2002), "Computer-integrated finite element modeling of human middle-ear", *Biomechanics and Modeling in Mechanobiology*, Vol. 4, pp. 190-199.
- [147] Thiebaut, C., and Lemonnier, D. (2002), "Three-dimensional modelling and optimisation of thermal fields in a human body during hypothermia", *International Journal of Thermal Sciences*, Vol. 41, pp. 500-508.
- [148] Bradley, C., and Pullan, A. (2002), "Application of the BEM in biopotential problems", *Engineering Analysis with Boundary Elements*, Vol. 26 (5), pp. 391-403.
- [149] Seo, H., and Magnenat-Thalmann, N. (2004), "An Example-Based Approach to Human Body Manipulation", *Graphical Models*, Vol. 66 (1), pp. 1-23.

- [150] Chu, T.M., Reddy, N.P., and Padovan, J. (1995), "Three-dimensional finite element stress analysis of the polypropylene ankle-foot orthosis: static analysis", *Medical Engineering & Physics*, Vol. 17, pp. 372-379.
- [151] Terzopoulos, D., and Fleischer, K. (1988), "Modeling inelastic deformation: viscoelasticity, plasticity, fracture", *Computer Graphics*, Vol. 22 (4), pp. 269-278.
- [152] Jansson, J., and Vergeest, J.S.M. (2002), "A discrete mechanics model for deformable bodies", *Computer-Aided Design*, Vol. 34 (12), pp. 913-928.
- [153] Bourguignon, D., and Cani, M.P. (2000), "Controlling anisotropy in mass-spring systems", *Proceedings of the 11th Eurographics workshop, Computer Animation and Simulation*, pp. 113-123.
- [154] McDonald, J. (2001), "On flexible body approximations of rigid body dynamics", *Proceedings of the WSCG, Plzen*.
- [155] Foster, N., and Fedkiw, R. (2001), "Practical animation of liquids", *Proceedings of the ACM SIGGRAPH, Los Angeles*, pp. 23-30.
- [156] Premože, S., Tasdizen, T., Bigler, J., Lefohn, A., and Whitaker, R.T. (2003), "Particle-based simulation of fluids", *Proceedings of the Eurographics, Granada*.
- [157] Helsen, J., Heirman, G., Vandepitte, D., and Desmet, W. (2008), "The influence of flexibility within multibody modeling of multi-megawatt wind turbine gearboxes", *Proceedings of the ISMA International Conference on Noise & Vibration Engineering*, pp. 2045-2071.
- [158] Li, S., and Liu, W.K. (2002), "Meshfree and particle methods and their applications", *Applied Mechanics Reviews*, Vol. 55 (1), pp. 1-26.
- [159] Tsukanov, I., and Shapiro, V. (2002), "The architecture of SAGE - a mesh-free system based on RFM", *Engineering with Computers*, Vol. 18 (4), pp. 295-311.
- [160] Zhang, Y., Prakash, E.C., and Sung, E. (2004), "Face alive", *Journal of Visual Languages and Computing*, Vol. 15, pp. 125-160.
- [161] Nedel, L.P., and Thalmann, D. (2000), "Anatomic modeling of deformable human bodies", *The Visual Computer*, Vol. 16, pp. 306-321.
- [162] Cerezo, E., Pina, A., and Serón, F.J. (1999), "Motion and behaviour modelling: state of the art and new trends", *The Visual Computer*, Vol. 15, pp. 124-146.
- [163] Zhang, L., Liu, W.K., Li, S., Qian, D., and Hao, S. (2002), "Survey of multi-scale meshfree particle methods", *Lecture Notes in Computer Science Engineering*, Vol. 26, pp. 441-458.
- [164] Moenning, C., Mémoli, F., Sapiro, G., and Dodgson, N.A. (2004), Meshless geometric subdivision. IMA Preprint Series # 2001, Institute for Mathematics and its Applications, Univ. of Minnesota., Minnesota.
- [165] Meiling, Z., Yufeng, N., and Chuanwei, Z. (2004), "A new coupled MPLG-FE method for electromagnetic field computations", *Proceedings of the IEEE International Conference on Computational Electromagnetics and Its Applications*, pp. 29-32.
- [166] Törngren, M., Henriksson, D., Årzen, K.R., Cervin, A., and Hanzalek, Z. (2006), "Tool supporting the co-design of control systems and their real-time implementation: current status and future directions", *Proceedings of the IEEE Conference on Computer Aided Control Systems Design, Munich*, pp. 1173-1180.
- [167] Assanis, D., Filipi, Z., Gravante, S., Grohnke, D., Gui, X., Louca, L., Rideout, G., Stein, J., and Wang, Y. (2000), "Validation and use of Simulink integrated, high fidelity, engine-in-vehicle simulation of the international class VI truck", *SAE Transactions*, Vol. 109 (3), pp. 384-399.

- [168] Bennett, S. (1996), "A brief history of automatic control", *IEEE Control Systems Magazine*, Vol. 16 (3), pp. 17-25.
- [169] Brown, F.T. (2001), *Engineering system dynamics - a unified graph-centered approach*, Marcel Dekker, Inc., New York.
- [170] Vahid, F., and Givargis, T. (2002), *Embedded systems design - A unified hardware/software introduction*, John Wiley & Sons, Inc., New York.
- [171] Rosenbaum, D.A. (1991), *Human motor control*, Academic press, San Diego.
- [172] Stelmach, G.E. (1982), "Information-processing framework for understanding human motor behavior", in: *Human motor behavior - an introduction*, pp 63-91, Kelso, J.A.S. (Ed.), Lawrence Erlbaum Associates, London.
- [173] Wickens, C.D., and Hollands, J.G. (2000), *Engineering psychology and human performance*, Prentice-Hall, Upper Saddle River.
- [174] McRuer, D. (1980), "Human dynamics in man-machine systems", *Automatica*, Vol. 16, pp. 237-253.
- [175] Grafton, S.T., and Hamilton, A.F. (2007), "Evidence for a distributed hierarchy of action representation in the brain", *Human Movement Science*, Vol. 26, pp. 590-616.
- [176] Hicks, M.J. (2004), *Problem solving and decision making*, Thomson Learning, London.
- [177] Eysenck, M.W., and Keane, M.T. (2000), *Cognitive Psychology*, 4th ed., Psychology Press, Hove, UK.
- [178] Regan, D. (1997), "Perceptual motor skills and human motion analysis", in: *Handbook of human factors and ergonomics*, Salvendy, G. (Ed.), John Wiley & Sons, New York, pp. 174-218.
- [179] Abend, W., Bizzi, E., and Morasso, P. (1982), "Human arm trajectory formation", *Brain*, Vol. 105, pp. 331-348.
- [180] Atkeson, C.G., and Hollerbach, J.M. (1985), "Kinematic features of unrestrained vertical arm movements", *The Journal of Neuroscience*, Vol. 5 (9), pp. 2318-2330.
- [181] Lindsay, P.H., and Norman, D.A. (1972), *Human information processing - an introduction to psychology*, Academic Press, New York.
- [182] Miller, J. (1988), "Discrete and continuous information-processing approach models", *Acta Psychologica*, Vol. 67, pp. 191-257.
- [183] Jagacinski, R.J., and Flach, J.M. (2003), *Control theory for humans*, Lawrence Erlbaum Associates, Mahwah NJ.
- [184] Costello, R.G. (1968), "The surge model of the well-trained human operator in simple manual control", *IEEE Transactions on Man-Machine Systems*, Vol. 9 (1), pp. 2-9.
- [185] Sanders, A.F. (1980), "Stage analysis of reaction processes", in: *Tutorials in Motor Behavior*, Stelmach, G.E., Requin, J. (Eds.), North-Holland, Amsterdam, pp. 331-354.
- [186] Van der Auweraer, H. (2008), "Frontloading design engineering through virtual prototyping and virtual reality: industrial applications", *Proceedings of the TMCE*, Vol. 1, Izmir, pp. 39-52.
- [187] Van der Auweraer, H., De Oliveira, L., Da Silva, M., Herold, S., Mohring, J., and Deraemaeker, A. (2006), "A Virtual Prototyping Approach to the Design of Smart Structures Applications", *Proceedings of the ISMA International Conference on Noise and Vibration Engineering*, Leuven, pp. 273-284.
- [188] Alexik, M. (2000), "Modelling and identification of eye-hand dynamics", *Simulation Practice and Theory*, Vol. 8, pp. 25-38.

- [189] Multon, F., Nougaret, J.L., Arnaldi, B., Hégron, G., and Millet, L. (1999), "A software system to carry out virtual experiments on human motion", *Proceedings of the IEEE Computer Animation Conference*, pp. 16-23.
- [190] Veloso, A., Esteves, G., Silva, S., Ferreira, G., and Brandão, F. (2006), "Biomechanics modeling of human musculoskeletal system using Adams multibody dynamics package", *Proceedings of the 24th IASTED International Conference on Biomedical Engineering*, Innsbruck, pp. 401-407.
- [191] Park, W., Chaffin, D.B., Martin, B.J., and Yoon, J. (2008), "Memory-based human motion simulation for computer-aided ergonomic design", *IEEE Transactions on Systems Man and Cybernetics Part a-Systems and Humans*, Vol. 38 (3), pp. 513-527.
- [192] Yang, J.Z., Rahmatalla, S., Marler, T., Abdel-Malek, K., and Harrison, C. (2007), "Validation of predicted posture for the virtual human Santos (TM)", *Digital Human Modeling*, Vol. 4561, pp. 500-510.
- [193] Abdel-Malek, K., Yang, Y., Marler, R.T., Zhou, X., Patrick, A., and Arora, J. (2006), "Towards a new generation of virtual humans", *Int. Journal of Human Factors Modelling and Simulation*, Vol. 1 (1).
- [194] Gurney, K. (1997), *An introduction to neural networks*, UCL Press, London.
- [195] Martens, D. (1998), "Neural networks as a tool for the assessment of human pilot behaviour in wind shear", *Aerospace science and technology*, Vol. 3 (1), pp. 39-48.
- [196] Kim, J., and Hemami, H. (1998), "Coordinated three-dimensional motion of the head and torso by dynamic neural networks", *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 28 (5), pp. 653-666.
- [197] Reil, T., and Husbands, P. (2002), "Evolution of central pattern generators for bipedal walking in a real-time physics environment", *IEEE Transactions on Evolutionary Computation*, Vol. 6 (2), pp. 159-168.
- [198] Paulson, L.D. (2003), "Researchers automate the digital animation process", *Computer*, Vol. 36 (4), pp. 24-26.
- [199] NaturalMotion Ltd. (2006), *User Guide: endorphinTM dynamic motion synthesis*, Oxford/San Francisco.
- [200] Khoussianov, B., and Nerode, A. (2001), *Automata theory and its applications*, Birkhäuser, Boston.
- [201] Lee, D.T. (2002), "Evaluating real-time software specification languages", *Computer Standards & Interfaces*, Vol. 24, pp. 395-409.
- [202] Phillips, C.H.E. (1994), "Review of graphical notations for specifying direct manipulation interfaces", *Interacting with Computers*, Vol. 6 (4), pp. 411-431.
- [203] Tse, T.H., and Pong, L. (1991), "Examination of requirements specification languages", *Computer Journal*, Vol. 34 (2), pp. 143-152.
- [204] Gill, A. (1962), *Introduction to the theory of finite-state machines*, McGraw-Hill, New York.
- [205] Schruben, L. (1983), "Simulation modeling with event graphs", *Communications of the ACM*, Vol. 26 (11), pp. 957-963.
- [206] MacDonald, I.L., and Zucchini, W. (1997), *Hidden Markov and other models for discrete-valued time series*, Chapman & Hall, London.
- [207] Petri, C.A. (1962), "Fundamentals of a theory of asynchronous information flow", *Proceedings of the 1962 IFIP Congress*, Amsterdam, pp. 386-390.
- [208] Harel, D. (1987), "Statecharts: a visual formalism for complex systems", *Science of Computer Programming*, Vol. 8, pp. 231-274.

- [209] David, R., and Alla, H. (2005), *Discrete, continuous and hybrid Petri nets*, Springer, Berlin.
- [210] Francès, C., da Luz Oliveira, E., Costa, J., Santana, M., Santana, R., Bruschi, S., Vijaykumar, N., and de Carvalho, S. (2005), "Performance evaluation based on system modeling using Statecharts extensions", *Simulation Modelling Practice and Theory*, Vol. 13 (7), pp. 584-618.
- [211] Qin, J., and Xu, B. (2005), "Model checking for timed statecharts", *Lecture Notes in Computer Science*, Vol. 3731, pp. 261-274.
- [212] Cheng, A.M.K. (2002), *Real-time systems*, Wiley-Interscience, Hoboken.
- [213] Koznov, D.V., Kartashev, M., Gagarsky, R., Zvereva, V., and Barsov, A. (2004), "Roundtrip engineering of reactive systems", *Proceedings of the International Symposium on Leveraging Applications of formal methods (ISoLA)*, Paphos, Cyprus, pp. 343-347.
- [214] Kendall, I.R., and Jones, R.P. (1999), "An investigation into the use of hardware-in-the-loop simulation testing for automotive electronic control systems", *Control Engineering Practice*, Vol. 7, pp. 1343-1356.
- [215] Schnabel, M.K., Nenninger, G.M., and Krebs, V.G. (1999), "Konvertierung sicherer Petri-Netze in Statecharts - conversion of safe Petri nets into statecharts", *Automatisierungstechnik*, Vol. 47 (12), pp. 571-580.
- [216] Eshuis, R. (2005), "Statecharting Petri nets", Eindhoven University of Technology BETA Working Paper Series, 2005.
- [217] Bernardi, S., Donatelli, S., and Merseguer, J. (2002), "From UML sequence diagrams and statecharts to analysable Petri net models", *Proceedings of the ACM WOSP*, pp. 35-45.
- [218] Thompson, M.T., and Heimdahl, M.P.E. (1989), "An integrated development environment for prototyping safety critical systems", *Proceedings of the IEEE International Workshop on Rapid System Prototyping*, Clearwater Beach.
- [219] Mallach, E.G. (1975), "Emulator Architecture", *Computer*, Vol. 8 (8), pp. 24-32.
- [220] Drusinsky, D., and Harel, D. (1989), "Using statecharts for hardware description and synthesis", *IEEE Transactions on Computer-Aided Design*, Vol. 8 (7), pp. 798-807.
- [221] Yakovlev, A.V., Koelmans, M., Semonov, A., and Kinniment, D.J. (1996), "Modelling, analysis and synthesis of asynchronous control circuits using Petri nets", *Integration, the VLSI Journal*, Vol. 21, pp. 143-170.
- [222] Julia, S., and Valette, R.B. (2000), "Real time scheduling of batch systems", *Simulation Practice and Theory*, Vol. 8, pp. 307-319.
- [223] Orth, P., A., B., and Abel, D. (2005), "Rapid prototyping of sequential controllers with Petri nets", *Proceedings of the IFAC world Congress*, Prague.
- [224] Praehofer, H., and Pree, D. (1993), "Visual modeling of DEVS-based multi-formalism systems based on higraphs", *Proceedings of the winter Simulation Conference*, pp. 595-603.
- [225] Martel, A. (1998), "Application of ergonomics and consumer feedback to product design at Whirlpool", in: *Human factors in consumer products*, Stanton, N.A., Young, M.S. (Eds.), Taylor & Francis, London, pp. 75-90.
- [226] Christensen, S., Jørgensen, J.B., and Madsen, K.H. (1997), "Design as interaction with computer based materials", *Proceedings of the ACM-DIS 1997*, pp. 65-71.
- [227] Lenat, D.B., and Brown, J.S. (1984), "Why AM and EURISKO appear to work", *Artificial Intelligence*, Vol. 23, pp. 269-294.
- [228] Laird, J.E., Newell, A., and Rosenbloom, P.S. (1987), "SOAR: an architecture for general intelligence", *Artificial Intelligence*, Vol. 33, pp. 1-64.

- [229] Anderson, J.R., Matessa, M., and Lebiere, C. (1997), "ACT-R: A theory of higher level cognition and its relation to visual attention", *Human-Computer Interaction*, Vol. 12, pp. 439-462.
- [230] Kieras, D.E., Wood, S.D., and Meyer, D.E. (1997), "Predictive engineering models based on the EPIC architecture", *ACM Transactions on Human-Computer Interaction*, Vol. 4 (3), pp. 230-275.
- [231] Cutkosky, M.R., and Howe, R.D. (1990), "Human grasp choice and robotic grasp analysis", in: *Dextrous robot hands*, Venkataraman, S.T., Iberall, T. (Eds.), Springer, New York, pp. 5-31.
- [232] Bullock, D., and Grossberg, S. (1988), "Neural dynamics of planned arm movements: emergent invariants and speed-accuracy properties during trajectory formation", *Psychological Review*, Vol. 95 (1), pp. 49-90.
- [233] Morasso, P. (1981), "Spatial control of arm movements", *Experimental Brain Research*, Vol. 42, pp. 223-227.
- [234] Paulignan, Y., Frak, V.G., Toni, I., and Jeannerod, M. (1997), "Influence of object position and size on human prehension movements", *Experimental Brain Research*, Vol. 114, pp. 226-234.
- [235] Friedman, J., and Flash, T. (2007), "Task-dependent grasp selection of grasp kinematics and stiffness in human object manipulation", *Cortex*, Vol. 43, pp. 444-460.
- [236] Fogel, L.J., and Moore, R.A. (1968), *Modeling the human operator with finite-state machines* (NASA contractor report No. 1112), National Aeronautics and Space Administration, Washington DC.
- [237] Liu, A., and Salvucci, D.D. (2001), "Modeling and prediction of human driver behavior", *Proceedings of the 9th International Conference on Human-Computer Interaction*, Los Angeles CA, pp. 1479-14839.
- [238] Badler, N.I., Phillips, C.B., and Webber, B.W. (1993), *Simulating humans - computer graphic animations and control*, Oxford University Press, New York.
- [239] Badler, N.I., Palmer, M.S., and Bindiganavale, R. (1999), "Animation control for real virtual humans", *Communications of the ACM*, Vol. 42 (8), pp. 65-73.
- [240] Badler, N.I., Bindiganavale, R., Allbeck, J.M., Schuler, W., Zhao, L., and Palmer, M.S. (2000), "Parameterized Action Representation for Virtual Human Agents", in: *Embodied conversational agents*, Cassell, J., Sullivan, J., Prevost, S., Churchill, E. (Eds.), MIT Press, Cambridge, MA, pp. 256-284.
- [241] Carruth, D.W., Thomas, M.D., Robbins, B., and Morais, A. (2007), "Integrating perception, cognition and action for digital human modelling", *Lecture Notes in Computer Science* (4561), pp. 333-342.
- [242] Robbins, B., Carruth, D., and Morais, A. (2009), "Bridging the gap between HCI and DHM: the modeling of spatial awareness within a cognitive architecture", *Lecture Notes in Computer Science*, Vol. 5620, pp. 295-304.
- [243] Abdel-Malek, K., Yang, J.Z., Kim, J.H., Marler, T., Beck, S., Swan, C., Frey-Law, L., Mathai, A., Murphy, C., Rahmatallah, S., and Arora, J. (2007), "Development of the virtual-human santos (TM)", *Digital Human Modeling*, Vol. 4561, pp. 490-499.
- [244] Stanton, N.A., and Baber, C. (1998), "A systems analysis of consumer products", in: *Human factors in consumer products*, Stanton, N.A. (Ed.), Taylor & Francis, London, pp. 75-90.
- [245] Newell, A., and Simon, H.A. (1971), "Simulation of human thought", in: *Computer simulation of human behavior*, Dutton, J.M., Starbuck, W.H. (Eds.), Wiley and Sons, New York.

- [246] Hsia, P., Samuel, J., Gao, J., Kung, D., Toyoshima, Y., and Chen, C. (1994), "Formal approach to scenario analysis", *IEEE Software*, Vol. 11 (2), pp. 33-41.
- [247] Uchitel, S., Kramer, J., and Magee, J. (2004), "Incremental elaboration of scenario-based specifications and behavior models using implied scenarios", *ACM Transactions on Software Engineering and Methodology*, Vol. 13 (1), pp. 37-85.
- [248] Alspaugh, T.A., Richardson, D.J., and Standish, T.A. (2005), "Scenarios, state machines and purpose-driven testing", *Proceedings of the 4th International Workshop on Scenarios and State Machines*.
- [249] Glinz, M. (1995), "An integrated formal model of scenarios based on statecharts", *Lecture Notes in Computer Science*, Vol. 989, pp. 254-271.
- [250] Amyot, D., and Eberlein, A. (2003), "An Evaluation of Scenario Notations and Construction Approaches for Telecommunication Systems Development", *Telecommunication Systems*, Vol. 24 (1), pp. 61-94.
- [251] Elkoutbi, M., Khriiss, I., and Keller, R.K. (1999), "Generating user interface prototypes from scenarios", *Proceedings of the IEEE International Symposium on Requirements Engineering*, Limerick, pp. 150-158.
- [252] Khriiss, I., Elkoutbi, M., and Keller, R.K. (1999), "Automating the synthesis of UML StateChart diagrams from multiple collaboration diagrams", *Unified Modeling Language*, Vol. 1618, pp. 132-147.
- [253] Perlin, K., and Goldberg, A. (1996), "Improv: a system for scripting interactive actors in virtual worlds", *Proceedings of the SIGGRAPH*, pp. 205-215.
- [254] Cremer, J., Kearney, J., and Papelis, Y. (1995), "HCSM: a framework for behavior and scenario control in virtual environments", *ACM Transactions on Modeling and Computer Simulation*, Vol. 5 (3), pp. 242-267.
- [255] Hou, H., Sun, S., and Pan, Y. (2007), "Research on virtual human in ergonomic simulation", *Computers & Industrial Engineering*, Vol. 53 (2), pp. 350-356.
- [256] Tideman, M., van der Voort, M., and van Houten, F. (2008), "A new product design method based on virtual reality, gaming and scenarios", *International Journal on Interactive Design and Manufacturing*, Vol. 2 (4), pp. 195-205.
- [257] Horváth, I. (2004), "On some crucial issues of computer support of conceptual design: what to consider in order to be successful", in: *Product Engineering - eco-design, technologies and green energy*, Talaba, D., Roche, T. (Eds.), Springer, Dordrecht, pp. 123-142.
- [258] Chang, K.H., and Joo, S.H. (2006), "Design parameterization and tool integration for CAD-based mechanism optimization", *Advances in Engineering Software*, Vol. 37, pp. 779-796.
- [259] Jia, T., and Amirouche, F.M.L. (1989), "Optimum impact force in motion control of multibody systems subjected to intermittent constraints", *Computers & Structures*, Vol. 33 (5), pp. 1243-1249.
- [260] Mathworks (2007), *Stateflow and stateflow coder for use with Simulink - modeling, simulation, implementation*, The MathWorks, Inc., Natick.
- [261] Tiwari, A., Shankar, N., and Rushby, J. (2003), "Invisible formal methods for embedded control systems", *Proceedings of the IEEE*, Vol. 91 (1), pp. 29-39.
- [262] Orth, P., and Abel, D. (2006), "Rapid control prototyping petrinetzbasierter Steuerungen mit dem Tool NETLAB", in: *Automatisierungstechnik*, Vol. 54 (5), pp. 222-227.

- [263] Peng, S.S. (2004), "Ladder diagram and Petri-net-based discrete event control design methods", *IEEE Transactions On Systems, Man, And Cybernetics-Part C: Applications And Reviews*, Vol. 34 (4), pp. 523-531.
- [264] Kim, J.-H., Lee, N.Y., and Choi, J.-Y. (2006), "Formal Specification and Verification of PLC for Certification", *SIGBED Review (web version)*, Vol. 3 (4).
- [265] Zhang, J., Li, Q., Guo, Q., and Wang, Z. (2007), "A simulation method of controlled hybrid Petri nets based on Matlab Simulink/Stateflow", *Proceedings of the IEEE International Conference on Automation and Logistics*, Jinan, pp. 2432-2436.
- [266] Davidrajuh, R. (2008), "Developing a new Petri net tool for simulation of discrete event systems", *Proceedings of the 2nd Asia International Conference on Modelling & Simulation*, IEEE, Kuala Lumpur, pp. 861-866.
- [267] Speeter, T.H., Thompson, M.T., and Heimdahl, M.P.E. (1991), "Primitive based control of the Utah/MIT dextrous hand - an integrated development environment for prototyping safety critical systems", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 866-877.
- [268] O'Dwyer, A. (2000), "A summary of PI and PID controller tuning rules for processes with time delay. Part 1: PI controller tuning rules.", *Digital Control: Past, Present and Future of Pid Control*, pp. 159-164.
- [269] Arabyan, A., and Wu, F. (1998), "An improved formulation for constrained mechanical systems", *Multibody System Dynamics*, Vol. 2, pp. 49-69.
- [270] Branicky, M.S. (1995), "Universal computation and other capabilities of hybrid and continuous dynamical systems", *Theoretical Computer Science*, Vol. 138, pp. 67-100.
- [271] Holierhoek, J.G. (2007), "Rigid body modeling of wind turbines for aeroelastic stability investigations", *Proceedings of the AIAA Aerospace Sciences Meeting and Exhibit*, Reno.
- [272] Filla, R. (2005), "An event-driven operator model for dynamic simulation of construction machinery", *Proceedings of the 9th Scandinavian Conference on Fluid Power*, Linköping, Sweden.
- [273] Harel, D., Pnueli, A., Schmidt, J.P., and Sherman, R. (1987), "On the formal semantics of statecharts", *Proceedings of the Annual IEEE Symposium on Logic in Computing*.

INDEX

- action, 18, 73, **179**
 - entry -, 99
 - exit -, 99
 - transition -, 99, **182**
- ACT-R, 44, 45, 47, 57
- Adams, 27, 28, 30, 86, 87, 89, 90, 92, 94, 95, 96, 97, 108, 110, 114, 126, 127, 131, 132, 134, 137
- animation, 6, 13, 16, 17, 21, 25, 27, 28, 29, 30, 32, 33, 35, 41, 44, 46, 47, 87, 113, 115, 117, 141
- ANN. See artificial neural nets
- Anybody, 28, 48
- artefact, 2, 17, 43, 82, 113, **179**
- artificial neural nets, 40, 41
- automata. See finite automata
- behaviour, 3, 4, 8, 13, 17, 34, 49, 51, 52, 62, 64, 102, 110, **179**
 - human, reasoning model of, **18, 19**, 34, 38
- CAD, 13, 17, 20, 22, 23, 25, 27, 28, 32, 33, 35, 44, 52, 84, 86, 87, 140
- CADE. See computer-aided design engineering
- change relation, 8, 66, 68, 70, 72, 98, 99, **179**
- circumstances
 - ambient, 15, **179**
- climate simulation, 15
- cognitive architectures, 44, 45, 47, 51, 57
- computer-aided design engineering, 5, 7, 8, 59, 85, 136, 144, 153, 197
- condition values, 66, 67, 68, 69, 70, 73, 80
- construct, 3, 15, **179**
 - human interaction -. See human interaction construct
- control, 1, 2, 135, **179**
 - analogue, 37
 - artefact -, 36, 46, **179**
 - closed-loop, 14, 49
 - continuous, 36, 37, 39, 44, 47, 50, 78, 141
 - continuous vs. discrete, 36
 - digital, 37
 - discrete, 36, 39
 - embedded, 2, 27, 43, 47, 49, 54, 58, 59, 67, 82, 101, 108, 112, 113, 180
 - high-level, 38, 47, **180**
 - human. See control of human interaction
 - logical, 36, 41, 43
 - low-level, 38, 47, 137, **180**
 - of human interaction, 37, 51, 110
 - of simulations, 10, 14, 36, 39, 47, 86
 - of simulations, vs. simulation of control, 36
 - open-loop, 14
 - PID/PI/PD. See PID/PI/PD control
 - proportional, 39
 - scenario-based, 45
 - stages of, in human interaction, 38
- control behaviour, 18, 34, 40
- control signal, control value, control variable, 4, 37, 39, 41, 58, 63, 66, **69**, 70, 79, 81, 100, **179**
- control simulation, 23
- control value. See control signal
- control variable. See control signal
- control variable modifier, **69**
- coupling of simulations
 - loose / close, 29, 30, 33, 35, 51, 52, 54
- CPU time, 132
- DADS, 86
- decision making, 4, 6, 7, 38, 43, 44, 46, 47, 49, 50, 53, 54, 75, 78, 81, 82, 101, 116, 139, 140, 180
- delay, 83, **179**
- delayed transition, 75
- design concept, **59**, 65, 183, 184
- discrete flexibility, **32**, 53, 86
- duration, 103, 104, **179**
- duration value, **74**
- effector, 39, **179**
- embedded control. See control, embedded
- emulation, 43, 46
- end-delay event, **75, 179**
- end-delay transition, **75**
- entry action. See action
- event, 58, **68, 179**
- event-generating function, **73**
- evolutionary algorithms, 41, 44
- executable, **179**
- exit action. See action
- explicit event, **180**
- explicit logistic event, **99**
- external event, **68**
- eye-limb coordination, 38, 39, 47
- finite automata, 10, 23, **41**, 42, **43**, 44, 51, 67, 70, 75, 88, 89, 136, 140
- finite element method, 28, 39
- finite state machines. See finite automata
- forward dynamics, 95, 140
- game theory, 4
- genetic algorithms, 41

- grasping, 36, 37, 44, 107, 108, 109, 111, 112, 115, 137
- half space
 - infinitesimal, **60**, 92, 94
- hardware in the loop, 14
- HIC. *See* human interaction construct
- human in the loop, 14
- human interaction construct, 79, 82, 101, 102, **180**
- human-artefact interaction
 - reasoning model of, 54, 55, **56**, **121**
- human-artefact system, 180
- implicit event, **180**
- implicit logistic event, **99**
- instruction, **180**
- interaction, **180**
- internal event, **68**, 100
- invariants
 - of motion patterns, 38, 39, 44, 52, 90, 112, 141
- inverse dynamics
 - inverse kinematics, 33
- Jack*, 44
- kinetostatics, **26**, 27, 45
- LifeModeler, 28, 40, 48, 87
- LMS Virtual.Lab, 27, 30, 86, 87, **126**
- logical condition, 68
- logical construct, **180**
- logical constructs
 - set of, **66**, **67**
- logistic event, 66, 67, **69**, 83, **180**
- logistic values, 66, 67
- logistics layer, **65**, 66
- low-level logical control of human
 - motion
 - model of, 67, 78, 79, **81**, **89**, 137, 140
- Matlab. *See* Simulink
- meshing, 32, 35
- metabolism, 122
- meter events, **68**, **69**, 70, **73**
- meter signal, meter value, meter
 - variable, 58, 66, 70, 103, **180**
- meter value. *See* meter signal
- meter variable. *See* meter signal
- model, **180**. *See* simulation model
- motion capture, 40, 47, 141
- multibody simulation, 27
- multiphysics, 1, 4, 8, 24, 26, 29, 33, 35, 51, 53, 57, 62, 136
- neural nets. *See* artificial neural nets
- nucleus, 1, 8, 10, 11, 53, 55, 57, 58, 59, **60**, 61, 62, 64, 65, 71, 72, 75, 84, 85, 86, 87, 90, 91, 94, 97, 99, 119, 120, 131, 136, 137, 179, **180**, 181, 183, 184
- nucleus-based modelling, 8, 57, 59, **85**, **90**, 93, 97, 139, 140
- orientation
 - of a meter event, **73**, **100**
- particles, 32, 36, 53, 60, 61, 62, 85, 86, 88, 92, 93, 94, 95, 96, 109, 134, 139, 140, 184
 - boundary, **60**
 - internal, **60**
- Petri nets, 42, 43, 46, 47, 89, 126
- physically coupled pair, **59**, **181**
- physics
 - areas of, 6, 10, **17**, 19, 24, 26, 35, 50, 135, 136
- PID/PI/PD control, 37, 39, 40, 78, 95, 110, 140, 141
- PLD. *See* programmable logic device
- Preparation efforts indicator, **127**
- prescribed force relation, **63**, **94**
- prescribed torque relation, **63**, **94**
- prescribed velocity relation, **63**, **94**
- procedural logic, 22, 42, 44, 126
- procedure structure, 67, 78, **114**, **181**
- product, **181**
- programmable logic device, 36, 37, 43, 89
- recognition variable, **73**
- relational operator, 70
- relations, **181**
 - binary, **62**
 - unary, **62**
- remeshing. *See* meshing
- resource-integrated modelling, 8, **57**, 119, 120
- response execution, **38**, 39, 40, 43, 78, 82, 101, 102, 110, 112, 113, 133, 180, 181
- response execution layer, 101, **181**
- response selection, **38**, 39, 43, 78, 82, 101, 102, 110, 111, 112, 113, 115, 133, 180, 181
- response selection layer, 101
- response-execution primitives, **102**, **113**
- response-selection layer, 111
- robotics, 90
- Santos virtual human, 40, 45, 46, 49, 52
- scenario bundle, 1, **9**, 11, 55, 67, 75, 78, 81, 84, 102, **116**, 119, 121, 125, 127, 128, 136, 138, 140
- scenario layer, 101, 110, **181**
- scenarios, 1, 2, 6, 7, 9, 10, 11, 12, 13, 14, 36, 39, 45, 46, 47, 49, 51, 53, 54, 65, 79, 81, 88, 107, 109, 111, 115, 116, 121, 125, 127, 128, 129, 133, 136, 137, 139, **181**
 - in the strictest sense, 54
 - informal, 2, 5, 7, 53, 117
- SCS. *See* signal conversion specification
- signal. *See* control signal, meter signal
- signal conversion specification, **66**, 78, **80**, 83, 181
- signal flows of controlled interaction
 - simulation, 78

- SIL. See software in the loop
- Simpack, 27, 86, 126
- simulation, **13**, **181**
 - behavioural, 1, 4, 6, 8, 9, 13, 14, 15, 33, 34, 35, 48, 50, 51, 55, 58, 109, 120
 - continuous, **17**, 39, 43, 112
 - interactive, 5, 28
 - of control, 36, 47
 - of control vs. control of simulations, 36
- simulation experiment, 13, **14**
- simulation model, 3, 7, 13, 14, 15, 17, 20, 21, 23, 27, 28, 33, 39, 43, 48, 50, 52, 65, 66, 69, 70, 77, 78, 80, 81, 83, 84, 87, 88, 90, 105, 112, 114, 126, 127, 128, 131, 133, 134, 136, 138, 140, 183
- Simulation time indicator, **130**
- simulations
 - close and loose coupling of, 29, 51
- Simulink, 89, 114, 134, 137
- situations, **60**, 71, **181**
 - mapping to states, 72
- skinning, **29**, 31, 33
- software in the loop, 14, 36, 47
- source state
 - of a transition, **70**
- specification, **181**
- spring-damper, 21, 24, 32, 92, 94, **96**, 117, 137, 141
- start event, **72**, **74**, 103, 104
- start-delay event, **74**, 103, 104, **181**
- state entry action, **181**
- state exit action, **181**
- state machines. See finite automata
- state transition diagram, 42
- statecharts, 42, 43, 46, 47, 89, 97, 98, 126, 140, **182**
- Stateflow, 89, 97, 98, 100, 101, 110, 114, 127, 140
- states, 41, 68, **181**
 - mapping to situations, 72
- STD. See state transition diagram
- step function, 70, 71, 82, 127, 183
- stimulus, 3, 14, 37, 68, 73, 122
- stimulus recognition specification, **72**, **73**, 80, 103
- subcharting
 - of statecharts, 98
- surface patch, 60, 61
- surroundings, **182**
- synchronization of transitions, **99**
- target state
 - of a transition, **70**
- time signal, **74**
- timing specification, **72**, **74**, 103, See transition, 68, **70**, 128, 130, 133, **182**
- transition action, **182**. See action
- transition condition, **70**, **182**
- triggering event, **182**
- UML, 22, 46
- use, **182**
- use process, **182**
- use-process simulation system, **182**
- user, **182**
- virtual prototyping, 13, 43, 51, 113, 125
- waiting state, **74**

APPENDIX 1 – GLOSSARY

Individual entries are written in bold italic, and listed alphabetically.

References to other entries are underlined.

action : a purposeful intervention in order to cause changes within (or to influence behaviour of) a system. See also: transition action.

ambient circumstances : (*ignored in this thesis*) conditions in the surroundings considered constant or stationary in this thesis, e.g., temperature and humidity of the air.

artefact : man-made physical object, in my context a product or an object forming an element of the surroundings in a use process

artefact control : generation of signals that enforce the operation of actuators, described in a procedure structure.

behaviour : observable concrete (or virtual) manifestation of the operation of a system governed by natural (physical, biological, etc.) laws

change relation: relation between two situations that are subsequent in time, specified as new value assignments to a set of control variables that all refer to the same nucleus.

construct : purposeful organization of interrelated entities; can be a model, a specification or a combination of both.

control signal : signal specified by the designer to enforce the operation of one specific effector as an output of a logical construct and as an input of a simulation.

control value : instantaneous value of a control signal

control: Regulatory action, typically based on (or influenced by) input from sensors or from perception. See also: low-level control, high-level control, human control, artefact control.

delay : a time interval signifying latency in human control or artefact control with a start-delay event, a duration, and an end-delay event.

designer : the creator of product specifications or models, typically the user of a use-process simulation system.

duration (of a delay) : meter value specified by the designer with the goal of assigning a length to the time interval between a specific start-delay event and the corresponding end-delay event.

effector : muscle in a human or actuator in an artefact

end-delay event : event defined in the scs, occurring at the end of a delay.

event : notable occurrence of a change, having no duration and expressed by a pulse signal described by a Boolean expression. See also: triggering event, logic event, start-delay event, end-delay event, implicit event, explicit event.

executable : adjective expressing that something (e.g., a file, an instruction) is compilable for processing by a computer

execute : to compile for, and process by a computer

explicit event : event with a name specified by the designer, that has been defined in a logical construct.

hic : human interaction construct

high-level control : control described by specifications, which reflect instructions (for instance, a program to be executed by embedded software) or human-generated conjectures of behaviour (for instance, a designer's conjecture of conscious human decision-making).

human : (virtual) person who uses the product (of which the use is investigated through use-process simulation).

human control : generation of signals that enforce the operation of human muscles, described by the human interaction construct.

human interaction construct (hic) : logical construct in which control of human effectors (muscles) is specified and modelled as a statechart with three layers: the scenario layer, the response selection layer, and the response execution layer.

human-artefact system : system consisting of at least one human and one product (used by this human), and the surroundings of the human and the product

Implicit event : event generated automatically at a particular change in a statechart (e.g., a transition is taken) with a default name derived from the name of that change (e.g., in the case of a transition being taken, an implicit event with the same name as the event is generated).

instruction : executable command represented as a specification, not as a model.

interact : to be engaged in interactions.

interaction : action taking place in the human-artefact system during a time interval in a use process as demarcated by two transitions between situations.

logical construct : construct describing control of use process simulation in the form of a statechart.

logistic event: explicit or implicit event that has relevance only within a specific logical construct.

low-level control : control described by models, which reflect behaviour governed by natural laws (for instance, subconscious adjustments of human motions)

meter signal : continuous input signal of the scs, which it processes to generate input for logical constructs.

meter value : instantaneous value of a meter signal.

model : simplified virtual⁶⁷ representation of a system (human, artefact, surroundings or a combination of those) including a simulatable description of its behaviour.

nucleus : generic formulation of the logical and physical relations between at most two entities, assuming a particular situation in which the operation of

⁶⁷ Although, in general, models can be tangible, models discussed in this thesis are virtual

the nucleus is contemplated and supposed to occur

physically coupled pair : (virtual representation of) a physical manifestation of a nucleus including (part of) two physical objects connected by a physical relation in a given situation.

procedure structure : set of instructions defining the operation of effectors in an artefact (typically a product).

product : artefact designed by a designer for human use.

relation : a specific construct specifying how two things are associated, or how a thing is associated to itself.

response execution layer : statechart-based model of biologically determined low-level control of muscles, specifying quantitative changes in velocities of controlled body parts.

response selection layer : statechart-based model of biologically determined low-level control of muscles, specifying (i) the involved body parts, (ii) the required motions, and (iii) velocities of those motions as qualitative descriptions.

scenario : specification by a designer of a possible way for a human to apply high-level control to his interactions with a given product he or she is using in given surroundings.

scenario bundle : specification of a set of scenarios created by a designer to express his conjecture of human decision making in a use process.

scenario layer : layer of the human interaction construct containing the scenario bundle

scs : signal conversion specification

signal conversion specification (scs) : specification of operations to be performed on (i) the 'start simulation' command, (ii) meter signals, and (iii) start-delay events.

simulatable : if appropriately compiled for a computer, resulting in a simulation,

simulation : computational prediction of the behaviour of a system that is represented by models.

situation : a particular assumed or real-life physical set-up and a time interval with unchanging relations in a physically coupled pair.

specification : a prescriptive representation either as instructions or as another executable form describing actions performed by⁶⁸ a system.

start-delay event : explicit event specified by the designer as a transition action in order to define a delay.

state : form of existence of a system (or part of it) after a specific transition.

state entry action : transition action attached to all incoming transitions of a state.

state exit action : transition action attached to all outgoing transitions of a state

⁶⁸ A specification can also describe actions to be performed on a system, especially if it is a specification of how the system is manufactured. However, this interpretation falls outside the scope of the thesis

statechart : graphical logical representation introduced by David Harel in 1987 [208], and formalized in [273].

surrounding artefacts : artefacts forming surroundings in a use process and having their behaviour included in use process simulation.

surrounding nature (*ignored in this thesis*) : organisms, geological elements, and meteorological elements in the surroundings.

surroundings : part of the environment in which a human and a product interact during a use process, consisting of (i) surrounding artefacts, (ii) surrounding nature, and (iii) ambient circumstances.

transition : change of a system from one of its states to another specified state, that takes place if a specified triggering event occurs under specified transition conditions, that optionally effectuates specified change relations.

transition action : command attached to a transition, resulting in specified changes in control values or in the occurrence of an event (more specifically, a logistic event or a start-delay event).

transition condition : statement specified as a prerequisite for a transition.

triggering event : event that causes a transition if its transition conditions are satisfied.

use (noun) : application to a purpose; **(verb)** : to apply to a purpose

use process : a particular (goal-driven) sequence of interactions between a human, a product, and possible surrounding artefacts.

use-process simulation system : system capable of performing simulations of use processes in which effectors and actions are controlled

user : the term 'user' is avoided in this thesis because of its ambiguity, namely that it might refer either to the human who uses the designed product or to the designer who uses the use-process simulation system.

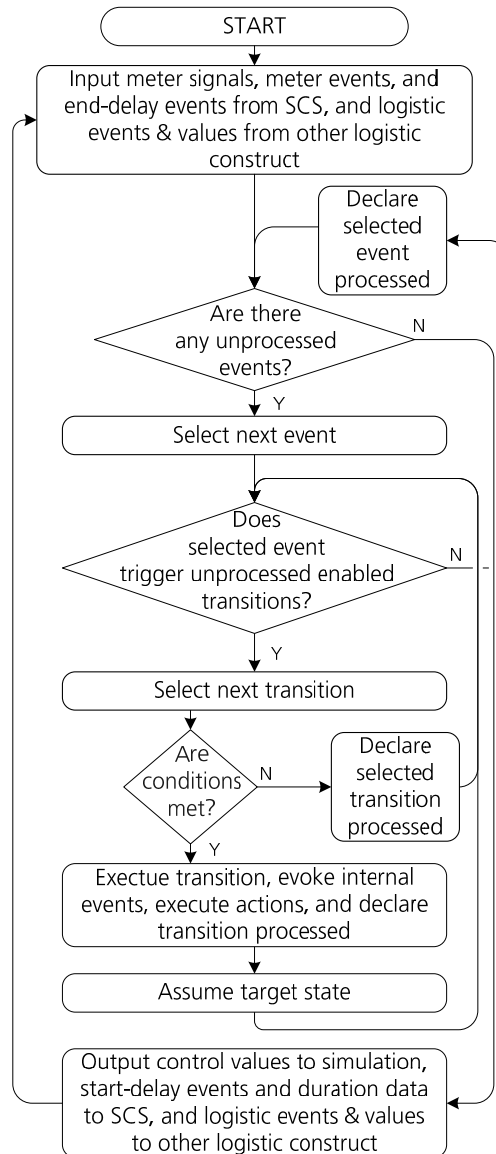
APPENDIX 2 – LIST OF SYMBOLS

- Enumerative indices (i, j, \dots) have been omitted for clarity.
- A set of attributes of objects in a design concept
- $B(N)$ behaviour of a nucleus
- C set of constraints on attributes, parameters, and descriptors of a design concept
- c step function describing changes in a control value
- $\dots(c)$... of the conventional simulation approach
- D set of descriptors of a situation
- d duration of a delay
- DC design concept
- e event
- e_{ext} external event
- e_{lgs} logistic event
- e_{met} meter event
- e_{td} delay-triggering event
- e_{sd} start-delay event
- F force
- f function (in particular, event-generating function)
- Fp prescribed force
- G behaviour generator function
- h event threshold
- HS infinitesimal half-space
- K control interval
- K_P proportional gain of a PI controller
- K_I integral gain of a PI controller
- k control variable modifier
- l influence descriptor (of an event on a control variable)
- L logistics layer
- M torque
- m meter value
- Mp prescribed torque
- N nucleus
- n number of investigated variations of a simulation model
- N_A number of activities required for simulation preparation
- N_T number of transitions occurring during a simulation
- N_S number of states assumed during a simulation
- N_M number of meter variables in a simulation model
- n_{mov} number of movable parts in a simulation model
- N_C number of signals connecting a control specification and a simulation model
- O set of pairs of objects in a nucleus or design concept
- o object
- P set of parameters describing physical relations
- p parameter / control value
- PEI preparation efforts indicator
- \mathbf{r} reference vector of an object in nucleus-based modelling
- r orientation of an event
- S situation in space and time
- $\dots(s)$... of the scenario bundle-based simulation approach
- STI simulation time indicator

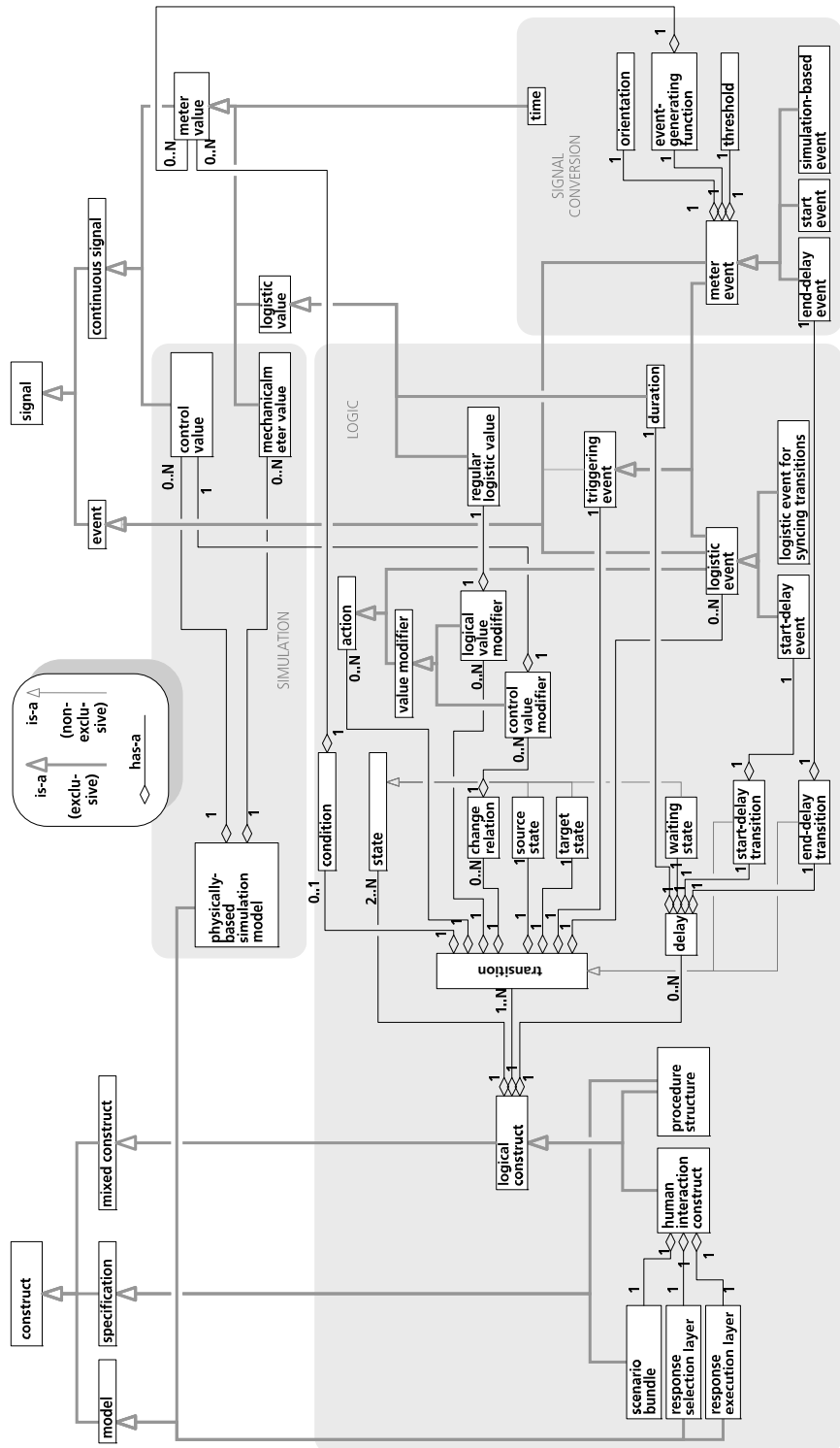
- T total CPU time
- t time
- t_{eval} hypothetical computational evaluation time for one time increment in physical simulation, and also for one time increment in execution of logical control of a simulation
- T_{ctrl} CPU time needed for execution of simulation control
- T_{sim} CPU time needed for simulation computation
- $\overline{t_{sim}}$ average simulation time between two consecutive transitions
- u logistic value
- v velocity
- vp prescribed velocity
- Z state
- Z_w waiting state
- γ logical condition
- Γ global reference frame in nucleus-based modelling
- Δt time increment
- δ change relation
- ε enabling of a transition
- ζ ratio between CPU time for control and CPU time for physically-based simulation
- \mathcal{A} set of logical constructs
- λ logical construct
- λ_ℓ model of low-level logical control of human motion
- λ_p procedure structure
- λ_s scenario bundle
- μ meter value appearing in the definition of an event-generating function
- ν condition value
- Ξ signal conversion specification
- ξ set of event specifications in the signal conversion specification (stimulus recognition events, start event and timing events)
- π particle
- Π particle cloud
- Π_s particle system
- Σ scenario
- τ transition
- τ_{sd} start-delay transition
- τ_{ed} end-delay transition
- ϕ set of physical relations in a design concept
- χ indicator function describing changes in a control value
- Ψ simulation layer
- ω angular velocity
- p prescribed angular velocity
- \diamond relational operator

APPENDIX 3 – PROCESSING

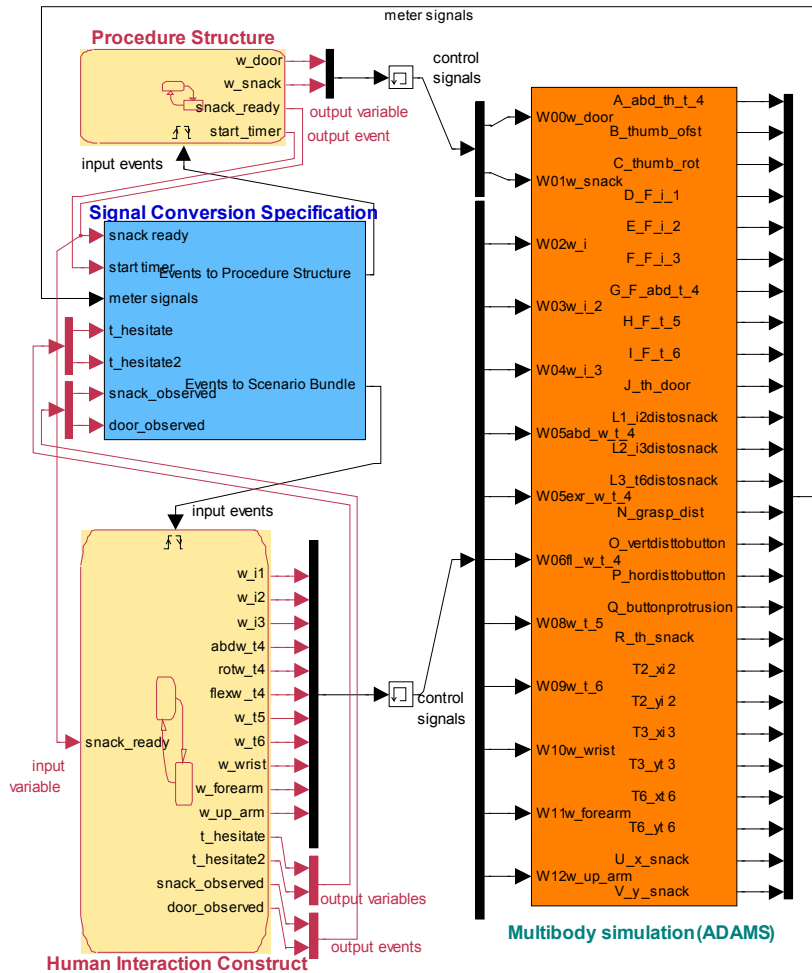
FLOWCHART OF THE HUMAN INTERACTION CONSTRUCT AND THE PROCEDURE STRUCTURE



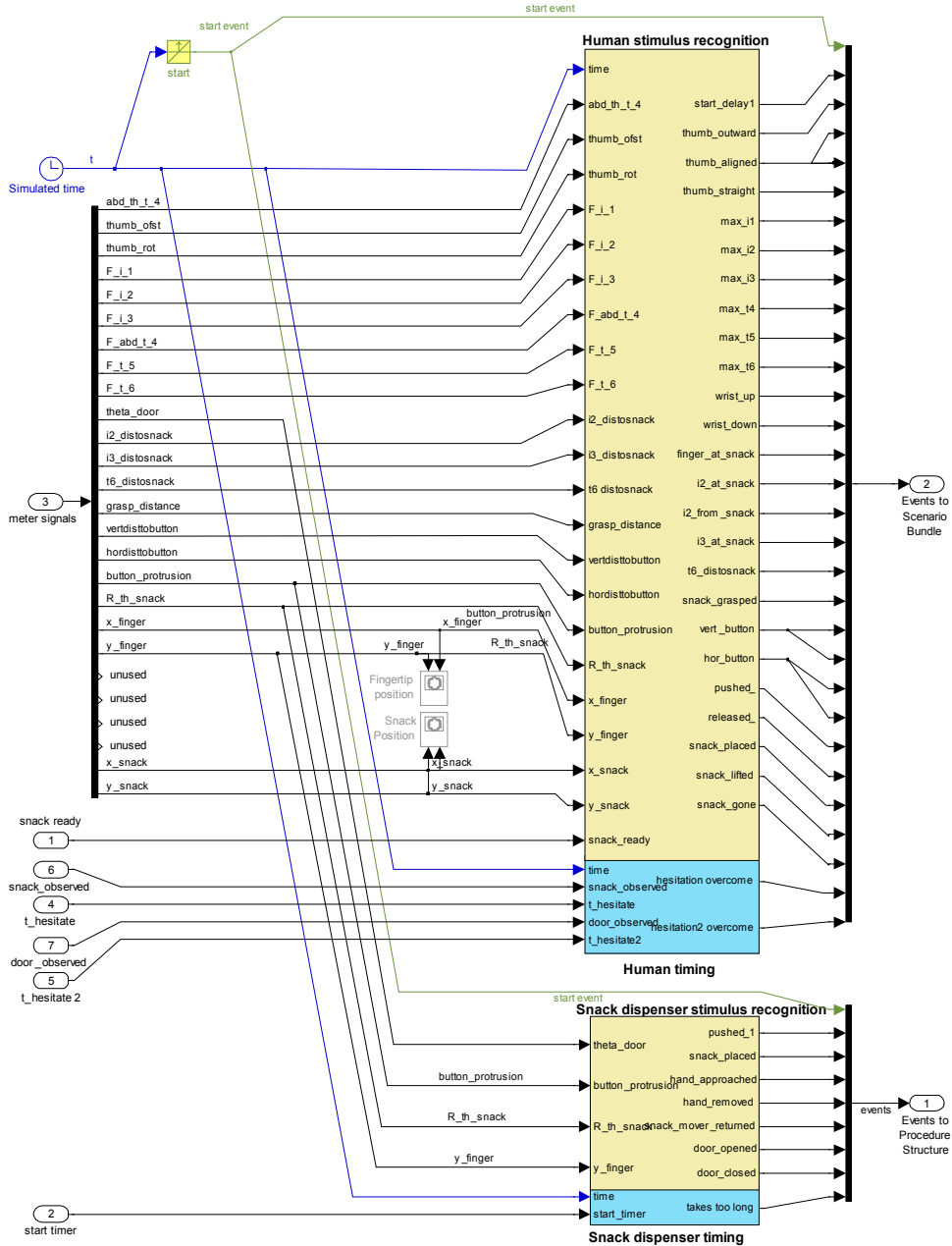
APPENDIX 4 – ENTITY- RELATIONSHIP DIAGRAM OF THE PROOF-OF-CONCEPT IMPLEMENTATION

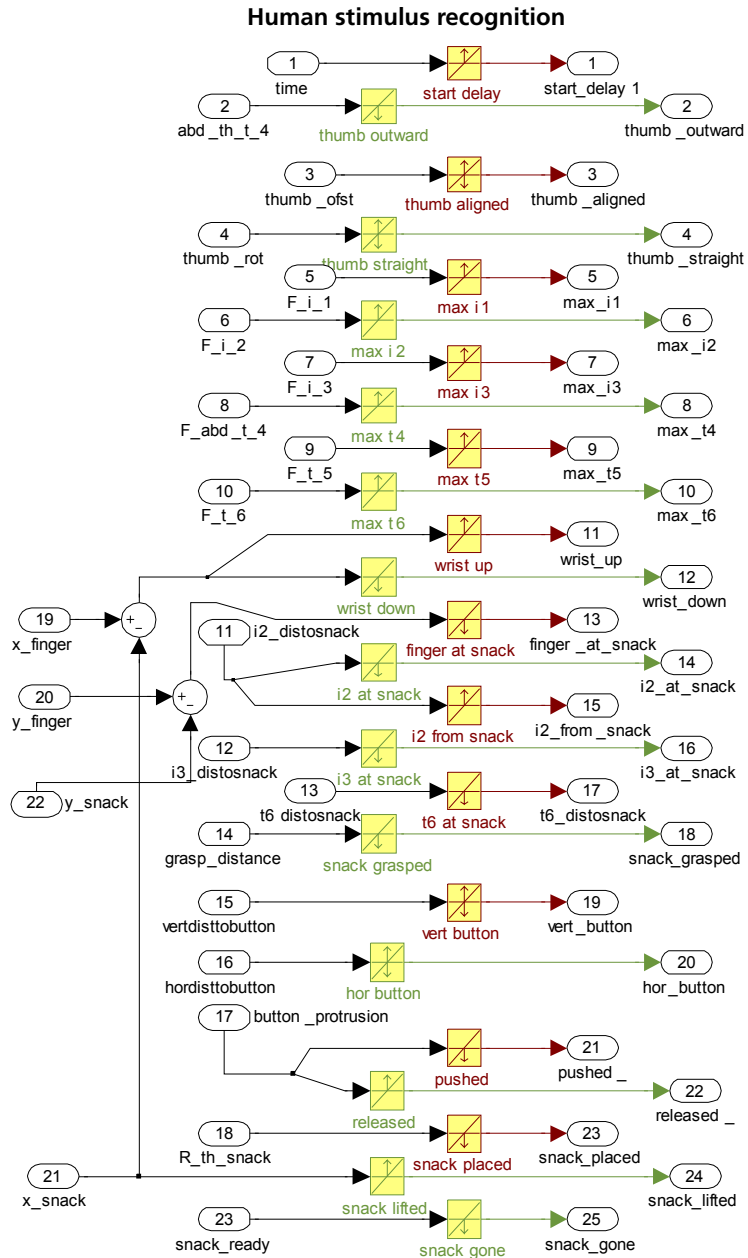


APPENDIX 5 – DETAILED SIMULINK CONSTRUCTS OF THE THIRD COMPOSITE SAMPLE CASE (SNACK DISPENSER)

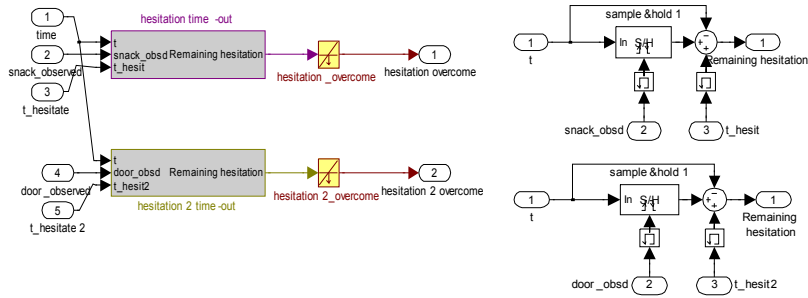


Signal conversion specification

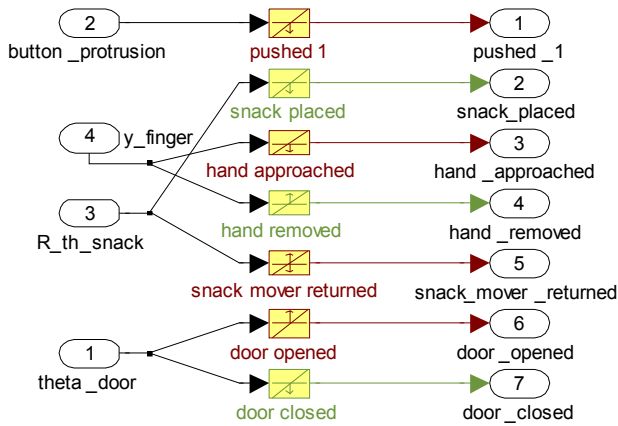




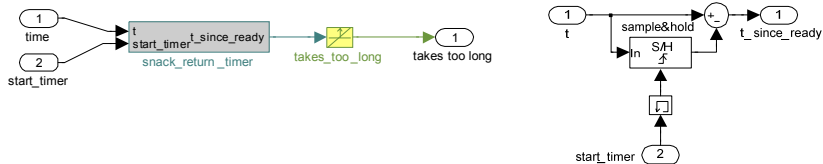
Human timing



Snack dispenser stimulus recognition

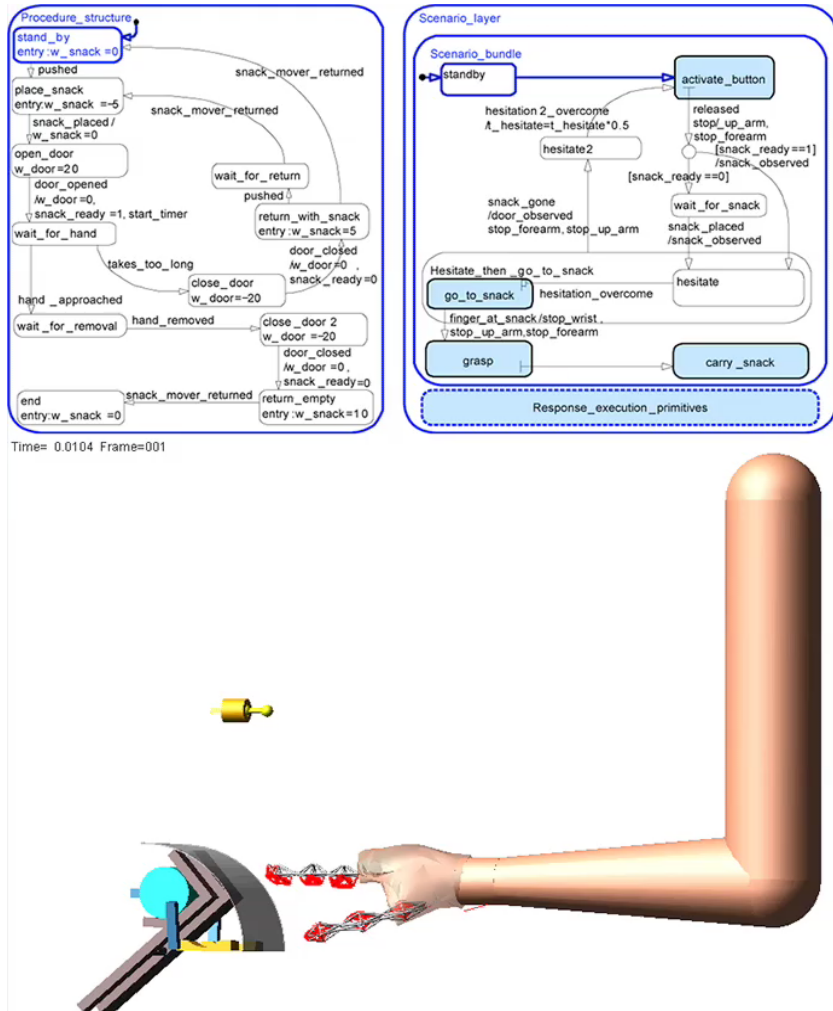


Snack dispenser timing



APPENDIX 6 – MOVIE CLIP

This movie gives an impression of how a system based on the approach described in this thesis might concurrently show an animation of a simulated use process together with the course through the scenario bundle and the procedure structure. The explanatory text has been added for demonstration purposes. Click on the image below to play the movie. (requires Apple™ Quick Time™, available from www.apple.com/quicktime/download)



ACKNOWLEDGMENTS

Although I have written much of this thesis in the first person singular, I would never have finished it without others inspiring me, helping me, supporting me, teaching me, or simply having to tolerate my state of mind. In fact, Professor Imre Horváth, my *promotor*, did all these things. Imre, I am very grateful for your efforts to make a scientist out of me – this thesis seems proof that you have finally succeeded. I hope to enjoy our deep intellectual debates and exchanges of ideas for years to come. To paraphrase a Dutch newspaper's slogan you have been, and still are, a *grindstone for my mind*.

I also owe a lot of thanks to all my other colleagues – current and past – in (CA)DE. Ernest van Breemen, you have been the perfect office mate. You always respected my need for concentration when I was writing, while at the same time offering distraction when I needed it, and you often helped me out with office software problems. And thanks for taking over the coordination of the io1030 course, a job you are doing much better than I ever could.

Zoltán Rusák gave me the idea for the final sample product, and the fruitful cross-fertilization between my work and his grasping project in its starting phase gave me new ideas. Joris Vergeest provided invaluable assistance in mathematical formalization issues and good advice in general for problems related to doing research. Adrie Kooijman managed to solve many of my computer problems, and also dug up nifty tips and tricks from the web, some of which invisibly found their way into this booklet. David Peck checked my propositions for proper English usage.

Considerable 'external' assistance was provided by Chris Verheul from Sayfield International, who helped me overcome the steep learning curve of the Adams multibody simulation package. Without his help, and the help of the people of the Adams discussion community on the web, I would probably have been lost.

Many other people have helped in addressing technical issues. For those not mentioned here, I am grateful to you nevertheless and I hope your contribution is somehow evident in the foregoing pages.

I took a lot of inspiration from my paranympths Elvin Karana and Magdalena Chmarra, who went through the process of finishing their PhDs (in completely different areas) earlier in 2009, and who contributed to this work from a friend's perspective. Elvin, I continue to enjoy our daily mutual updating on almost anything. Your contagious laughter always freshens up my mind, even when it's from somewhere up the corridor. Magda, the ever-expanding repertory of emoticons in your messages makes me smile. Your mighty *sernik* is so delicious that I had to ask for another one:)

And there's someone else who got her PhD before me - fifteen years minus six days, to be exactly. Christiane, although you also contributed in many practical and concrete ways, your love and warmth motivated me more than anything.

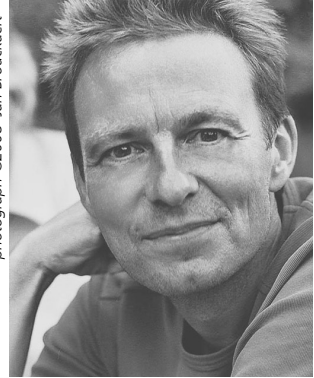
But I also have to mention your patience, even if it's not in your nature, because you were the one who had the most to endure from my long journey towards this achievement. I love you and I'm proud of you.

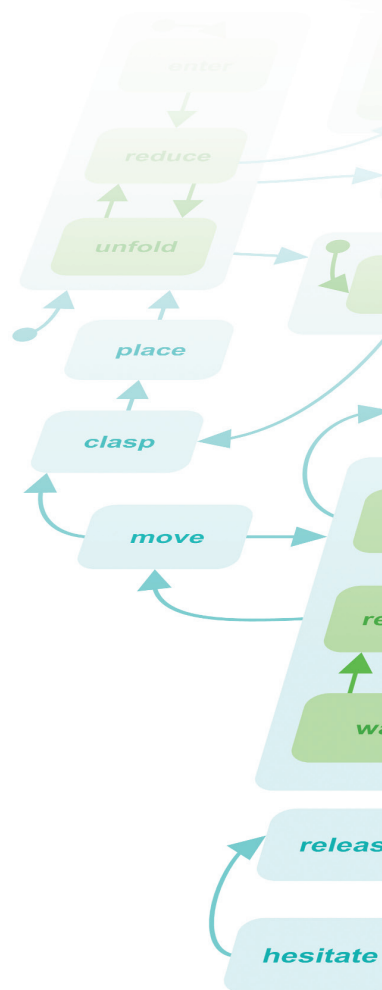
Wilfred van der Vegte
Delft, November 7, 2009.

CURRICULUM VITAE

Wilfred van der Vegte (1963) received a master's degree (equivalent) in mechanical engineering from Twente University in 1989. Subsequently he enrolled in the postgraduate course programme on design of consumer durables at Delft University of Technology, where he obtained his Master of Technological Design early 1992. As his final project he developed a working prototype of a graphics display for the blind at TNO Product Centre (later merged into the Institute of Industrial technology) in Delft, where he continued his career as an industrial designer and project manager until 1998. In his TNO years, he worked on a wide range of product development projects (varying from furniture to high-volume components for domestic appliances) and consultancy assignments in the field of *design for x* and *value engineering*. In 1998 he returned to the Faculty of Industrial Design Engineering at Delft University of Technology to become Assistant Professor in the Section of Computer-Aided Design Engineering, where he started doing research, and became engaged in teaching. Since then, the research work gradually developed towards the topic presented this thesis. His teaching activities include fundamental engineering topics, knowledge management in advanced design support, and coaching students' design projects and graduation projects.

photograph ©2008 Jan Brouckaert





ISBN 90-6562-097-4



9 789065 620972

Mitfred Vander Veegh Testing Virtual Use with scenarios

Propositions

accompanying the thesis

Testing virtual use with scenarios

by Wilhelm Frederik van der Vegte

1. Conceived human-product interactions within the domain of use processes can be formally specified as *scenario bundles*, which can be operationalized as control mechanisms for behavioural simulations (*this thesis*).
2. Existing simulation systems do not provide adequate means to simulate human-product interactions in the context of multifaceted use processes (*this thesis*).
3. Enhancing simulation approaches with the objective of increasing realism is a never-ending job (*this thesis*).
4. If an animation of a process looks convincing, it does not mean that it is realistic.
5. Just like non-SI units in the USA, Liberia, Burma and the UK, decimal commas should be abolished in continental Europe.
6. Intensive computer use promotes the assumption that the *undo* command can be applied to actions in everyday life.
7. It should be permitted to throw jars and bottles with lids, caps and corks into the glass recycling bin.
8. By only asking patients to return if the treatment is unsuccessful, physicians deny themselves a crucial source of feedback.
9. Dutch bars and restaurants should have fill lines on all glasses to protect consumers from fraud.

These propositions are considered opposable and defensible and have been approved as such by the supervisor Prof. dr. I. Horváth.

Stellingen

behorende bij het proefschrift

Testing virtual use with scenarios

door Wilhelm Frederik van der Vegte

1. Door geconcipeerde mens-productinteracties in gebruiksprocessen formeel te specificeren als *scenariobundels*, kunnen deze operationeel kunnen worden gemaakt als regelmechanismen van gedragssimulaties (*dit proefschrift*).
2. Bestaande simulatiesystemen bieden onvoldoende mogelijkheden om mens-productinteracties in gefacetteerde gebruiksprocessen te simuleren (*dit proefschrift*).
3. Het verbeteren van simulatiemethoden om meer realisme te bereiken is een eindeloze opgave (*dit proefschrift*).
4. Als een animatie van een proces er overtuigend uitziet betekent dat nog niet dat deze realistisch is.
5. Evenals de niet-SI-eenheden in de Verenigde Staten, Liberia, Birma en het Verenigd Koninkrijk dient de decimale komma in continentaal Europa te worden afgeschaft.
6. Veelvuldig computergebruik leidt tot de veronderstelling dat het commando 'maak ongedaan' ook kan worden toegepast op handelingen in het dagelijks leven.
7. Het dient te worden toegestaan om potten en flessen inclusief deksels, doppen en kurken in de glasbak te werpen.
8. Door patiënten alleen te vragen om terug te komen als de behandeling niet werkt, ontszeggen artsen zich een belangrijke bron van terugkoppeling.
9. Om consumenten tegen bedrog te beschermen dienen glazen in de Nederlandse horeca te worden voorzien van maatstreepjes.

Deze stellingen worden oponeerbaar en verdedigbaar geacht en zijn als zodanig goedgekeurd door de promotor Prof. dr. I. Horváth.