TU Delft

Delft University of Technology

Designing context-aware systems: A method for understanding and analysing context in practice

van Engelenburg, Sélinde; Janssen, Marijn; Klievink, Bram

DOI
10.1016/j.jlamp.2018.11.003

Publication date
2018

Document Version
Final published version

Published in
Journal of Logical and Algebraic Methods in Programming

Citation (APA)
van Engelenburg, S., Janssen, M., & Klievink, B. (2018). Designing context-aware systems: A method for understanding and analysing context in practice. *Journal of Logical and Algebraic Methods in Programming*, *103*, 79-104. https://doi.org/10.1016/j.jlamp.2018.11.003

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Journal of Logical and Algebraic Methods in Programming 103 (2019) 79–104

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Designing context-aware systems: A method for understanding and analysing context in practice

Sélinde van Engelenburg *, Marijn Janssen, Bram Klievink

Delft University of Technology, Jaffalaan 5, Delft 2628BX, The Netherlands

ARTICLE INFO

Article history:
Received 19 February 2018
Received in revised form 19 October 2018
Accepted 8 November 2018
Available online 15 November 2018

Keywords:
Context
Context awareness
Context elements
Design
Requirements engineering
Business-to-government information sharing

ABSTRACT

Context-aware systems are systems that have the ability to sense and adapt to the environment. To operate in large-scale multi-stakeholder environments, systems often require context awareness. The context elements that systems in such environments should take into account are becoming ever more complex and go beyond elements like geographic location. In addition, these environments are themselves so complex that it is hard to determine what parts of them belong to the relevant context of a context-aware system. However, insight into what belongs to this context is needed to establish what the design of a context-aware system should be to meet its goal. The ambiguity of what belongs to context in these complex organizational environments causes the design process to become either inefficient or less effective. In this paper, we provide a method to identify what elements of the environment are relevant context and to then base the design on this insight. The proposed method consists of three steps: 1) getting insight into context, 2) determining what components are needed to sense and adapt to context, and 3) determining the rules for how the system should adapt in different situations. To reduce ambiguity by organizations, the method requires a more specified definition of context than the ones in current literature, which we also provide in this paper. In addition to reducing ambiguity, the highly structured way in which the components and rules are derived from insight into context provides a way to further deal with the high complexity of the context. The method was applied for the development of a context-aware system for business-to-government (B2G) information sharing in the container shipping domain. Information sharing in this domain is highly complex, as legislation, many stakeholders, and a mix of cooperation and competition result in a highly complex environment. The development of this B2G information sharing system thus provides an example of how the method can be used to develop a context-aware system in a highly complex environment.
© 2018 The Authors. Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Technological developments, such as big data and the Internet of Things (IoT), result in more and more software applications taking the context into account. There are plenty of examples of developments that require such systems. A typical and traditional example in the literature is that of a context-aware tour guide. Such tour guides provide information to their users that is relevant considering, for instance, their location or personal preferences [1]. Another example are devices that

* Corresponding author.
E-mail addresses: S.H.vanEngelenburg@tudelft.nl (S. van Engelenburg), M.F.W.H.A.Janssen@tudelft.nl (M. Janssen), A.J.Klievink@tudelft.nl (B. Klievink).

https://doi.org/10.1016/j.jlamp.2018.11.003
2352-2208/© 2018 The Authors. Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

are used to maintain balance on a smart energy grid [2]. This requires the monitoring of energy points in houses [2]. This contextual information might then be used to coordinate the charging of electronic vehicles [2,3]. To achieve its purpose, such a system should interact with a variety of systems that monitor energy consumption and that switch on and off the charging of the vehicles. Furthermore, the system should be designed such that it meets the requirements of a variety of businesses, families and other parties that use and provide energy and information. In addition, it needs to be able to maintain a balance under a variety of circumstances, such as a heat wave, which leads to high energy consumption. The system thus requires taking into account the context. At the same time, the environment of the system is highly complex.

All in all, the types of opportunities enabled by developments such as the IoT and big data are growing and they often involve the sensing of contextual information. For example, IoT offers the opportunity to monitor almost every link in a supply chain and to adapt to changes in the market fast [4,5]. Such systems that sense and adapt to context are context-aware [6,7]. The environments in which these systems operate are often highly complex.

In 1994, Schilit and Theimer [8] were among the first to introduce the term 'context-aware' in relationship to computing. An overview of context-aware systems by Hong, Suh and Kim [6] shows that context awareness involves acquiring, sensing or being aware of context, as well as adapting to it or using it. Correspondingly, according to Dey and Abowd [9] context awareness falls into two categories, namely using context and adapting to context.

The environment in which a context-aware system exists can be viewed as open and infinite. However, when designing a system, only part of this environment is relevant to take into account, i.e. the context of the system. The identification of what parts of the environment are parts of the relevant context is non-trivial, as the environment can be highly complex and involve a high number of elements that are possibly relevant. Therefore, a method is needed to investigate the context of context-aware systems in complex environments and to determine what is relevant to take into account in their design.

The need for such a method originated from practice. In the international container-shipping domain, new systems supporting business-to-government (B2G) information sharing are required [10]. The main need for such a method originates from the high number of elements that possibly could be taken into account. There are several factors that make this environment highly complex. First of all, the idea that is currently dominant in the domain, is to support business-to-business (B2B) information sharing such that customs can reuse this data and piggy back on it (see e.g., [11–14]). This means that the system should support B2G information sharing, as well as B2B information sharing in a supply chain.

A supply chain is a massive set of tangled branches [15]. Supply chains include a high variety of businesses having different interest and a variety of legacy systems. These businesses vary on the type of services they provide (i.e., primary or secondary), their economic activity, level of technical capability and interdependencies with others, amongst others [16–19]. The information needs differ per organization and for high quality information the companies are dependent on each other. Those who make profit from selling data might not benefit from freely sharing it and they thus have no incentive to do so. The same business might perform different functions in different supply chains or within the same supply chain. In addition, often businesses authorise others to act on their behalf (e.g., customs broker).

Not only the businesses involved in the information sharing vary considerable. Different types of goods might be shipped as well, including high value, perishable, or dangerous goods. Furthermore, information sharing will often be international, which means that it is governed by legislation from a variety of sources. The information that is shared also can be of different types. There is IoT data from sensors, such as temperature or GPS location. In addition, businesses might share various contracts, invoices and other documentation with each other. Furthermore, they share various declarations with customs, such as Entry Summary Declarations and import declarations. In addition, a number of systems might be involved in information sharing and these might vary as well, for example, email, phone, and electronic data interchange (EDI) [20,21].

This complexity requires a context-aware system to support information sharing. It is likely that in different situations the information sharing needs to be supported in a different way by the system. Many of the things above could have an impact on whether and with whom information should and can be shared and in what way.

At the same time, this complexity makes it difficult to establish exactly what belongs to the context. It is difficult for many of these elements to see at first sight whether they should be taken into account as context in the design of the B2G information sharing system to sense and adapt to. A method is needed to either avoid the making (of unrealistic) assumptions, or spending a lot of resources on investigating each element. The former would make the design process less effective, while the latter would make the design process less efficient. Instead, we developed a new method for investigating context.

A detailed understanding of the context is needed to understand what a context-aware system should sense and adapt to. The involvement of, for example, many actors with different and even opposing requirements and various legislations based on the context, means that it is not easy to determine which elements in the context should be sensed and adapted to by the system. Identifying what belongs to the context requires an identification of the situations in which the system needs to operate and the balancing of the requirements of the stakeholders in those situations. In complex large-scale multi-stakeholder environments, the variety of these situations and stakeholders is larger.

At the same time, what belongs to the relevant context influences what the design of a context-aware system should look like. That is, it determines what sensors are needed to gather context information and what adaptors are needed to adapt to the context. It also determines what rules the system should incorporate for adapting to different situations based on the context information obtained from the sensors. This means that a thorough investigation of the context should be part of the integral design process for such systems.

Having a method to easily and systematically decide what belongs to context can help to improve the efficiency of the design process. Furthermore, having a method to gather and structure knowledge of the relevant context and systematically derive the design of the system from that, might improve effectivity.

The current literature on designing context-aware systems does not provide a method to systematically investigate context and base the design of a context-aware system on this (see section 2). In this research, we address this gap in knowledge by proposing a method that can be used to investigate context in a structured manner, to clearly distinguish irrelevant elements in the environment from relevant context elements and to base the design of a context-aware system directly on this insight.

In this paper, we use two systems as running examples, viz. a context-aware tour guide and a context-aware B2G information sharing system. The example of the tour guide will conform with the idea that most readers have of what a traditional context-aware system is, as there is quite some work on it [1]. It provides an easy to understand and familiar example. However, the method will be most useful in more complex cases, such as that of the B2G information sharing system. The development of such a system was what initially motivated the development of the method. The complexity is mainly in the high number of things that could belong to context. For the examples, we focus on what elements we found to be part of the context and how we identified them. It is impossible to show all elements that would likely have been taken into consideration fruitlessly without the use of the method. Therefore, we cannot use this example to show the full complexity of the domain. Instead, we use it to illustrate how the method can be used in practice.

This paper is structured as follows. In the following section, we discuss related work on context-aware systems. We then discuss our methods in section 3. Context can only be investigated effectively and efficiently when it is possible to easily decide what does and what does not belong to context. This requires a definition of context that makes clear what distinguishes relevant elements in the environment from irrelevant ones. In section 4, we provide a more specified definition of context. This definition is the basis for the gathering and analysis of data that is necessary to get insight into context in step 1 of our method. The method itself is presented in section 5. We provide a detailed discussion of what a designer should do to perform each step. Furthermore, we discuss why we believe that this is the appropriate course of action for the designer in that step. In section 6, we draw conclusions and discuss the implications of the use of the proposed method. We also describe how future work could help to further evaluate the method.

2. Related work

There is much related research available. However, this research is not focused on designing context-aware systems in the highly complex environments we discussed in the introduction. First, we discuss the work on designing context-aware systems. Then, in section 2.2, we discuss the work on requirement engineering in designing context-aware systems. Insight into context does not only need to be obtained during the design process and translated into design. The context-aware system also needs to sense context and adapt to it at runtime. Therefore, context needs to be modelled in the system as well. In section 2.3, we discuss some of the work on representing context information and rules for adapting to context. Determining what belongs to the relevant context of context-aware systems requires an unambiguous and easy to understand criterion for deciding this, as most actors are non-experts and have limited technological knowledge. Such a criterion should be derived from a clear definition of context. We discuss the related work on definitions of context in subsection 2.4. We provide our own definition of context in section 4.

2.1. Guidelines, tools and frameworks for developing context-aware systems

In 2009, Hong, Suh and Kim [6] found that about 5.5% of the papers on context-aware systems provide guidelines for development. A portion of this work, and more recent similar work, focuses on solving issues in a specific application domain or with specific types of sensors (see e.g. [22,23]). The majority of the work is on providing technical tools, frameworks and infrastructures that a designer can use to build context-aware applications (see e.g. [24–30]). In this 'infrastructure-centred approach' there is an assumption that the complexity of developing the systems can be reduced by using an infrastructure that can gather, manage and distribute context information [31]. These tools and frameworks for designing context-aware systems can be quite useful to designers, helping them to elaborate the technical details and create the system. However, this assumes that the context is investigated.

In a more recent survey, Alegre, Augusto and Clark [32] provide an overview of methods for engineering context-aware systems. An analysis of the described methods shows that they do not include the investigation of context as an explicit and fundamental stage in the design process. In concordance, Alegre, Augusto and Clark [32] conclude that the work does not include techniques and tools for understanding the context.

The research by Alegre, Augusto and Clark [32] does show that insight into context is important. On the basis of the results of a questionnaire completed by 750 researchers, they determine that among the most important features of a method for developing context-aware systems is the ability to represent situations in which the system should adapt in order to better understand them [32]. On the basis of an analysis of the literature, they state the following: *"All context information modelling and reasoning techniques need to enable the situation representation, but there is no support for understanding the situations and the contexts that they are going to be represented, stemming from the requirements."* [32, p. 24]. Our method, in

which getting insight into context is a fundamental stage in the design process, thus addresses a problem that is important to researchers involved in the design of context-aware systems.

2.2. Context awareness, design and requirements engineering

In their work on context-aware services, Finkelstein and Savigni [33] make a distinction between fixed goals and requirements. They state that goals are fixed objectives and requirements are volatile and can be influenced by context [33]. Due to their dependence on context, the functional requirements in the case of a context-aware system can be viewed as conditional. For example, 'filtering pricing information from the data' could be a requirement that needs to be met under the condition that a certain situation happens at runtime, e.g., when a competitor would get access otherwise.

This conditionality requires situations to be found in which the functional requirements for the system are different. This insight into context is necessary to establish what adaptors are needed to fulfil the functional requirements in those situations. Furthermore, relating these situations to the functional requirements is necessary to ascertain according to what rules the system should adapt. In addition, it needs to be determined what elements in those situations need to be sensed in order to identify the situation the system is in at runtime.

In general, the requirements engineering process does not seem to provide a way to get the insight into context that is needed to develop context-aware systems. Nuseibeh and Easterbrook [34] describe the requirements engineering process as consisting of context and groundwork, eliciting requirements, modelling and analysing requirements, and communicating requirements. The stage of context and groundwork is viewed as preparation and is used to determine the feasibility of the project and to select methods for further development [34]. It thus does not involve the systematic investigation of context that we need. The stage of requirements elicitation involves identifying stakeholders and goals [34]. This also does not involve linking situations to functional requirements.

The work specifically on requirements engineering in light of context awareness is limited. The notion of context-aware systems is often not mentioned explicitly in this work. Instead, the work discusses, for example, context-aware services, dynamic adaptive systems or self-adaptive systems [33,35,36]. This literature does, however, acknowledge the importance and different type of relationship between context and requirements in the case of context awareness [33,35,37]. In some cases, it even goes as far as viewing adaptation as requirements engineering by the system itself or as requiring requirement awareness by the system [35,36].

Even though the existing work appreciates the complexity of requirements engineering in the case of context awareness and the need to have insight into context, it does not provide a way to systematically get this insight. Instead, it proposes the use of existing usability methods or interviews, for instance [37,38]. Of course, such techniques might be useful in gathering the appropriate data necessary for getting insight into context. Nevertheless, these methods do not provide the structure necessary to investigate context in the case of large-scale multi-stakeholder environments. More specifically, they do not provide a way to easily decide what belongs to the context, and this is necessary to ensure efficiency.

Requirements engineering is part of a larger process of designing systems. According to Hevner [39], design science consists of a relevance cycle, rigour cycle and a design cycle. The connection to the environment made in the relevance cycle is to ensure that the design problem is relevant, that the artefact is usable in practice and that it actually offers a solution to the problem [39,40]. This cycle involves determining the requirements for the system and field testing of the system [39].

The distinctive property of context-aware systems is that they sense and adapt to context to deliver a solution to the design problem. Designing such systems thus involves determining what the system needs to sense, what adaptations it needs to make, and in response to what sensor information. Context awareness mainly impacts the design process in the relevance cycle, as this is where the connection to the environment is made.

However, for context-aware systems, an additional connection needs to be made to determine what conditions should be included for their conditional requirements. In other words, it needs to be determined what design could deliver the solution to the problem in different situations. When such a connection is not made, then the context taken into account is based on assumptions of the designer. These assumptions could be checked to some extent in the relevance cycle by determining whether a solution to the problem is found. However, this checking is indirect and thus leads to an inefficient design process. Furthermore, this way of testing makes it very hard, if not impossible, to rule out that parts of the context are included that do not help in solving the problem. This can lead to a design of a system that is needlessly complex.

Peffers et al. [41] describe several activities that are common in design science research, viz. 1) problem identification and motivation, 2) define the objectives for a solution, 3) design and development, 4) demonstration, 5) evaluation, and 6) communication. To ensure that the problem, objectives and solution are relevant, a connection to the environment needs to be made in the design process during activity 1 and 5. In activity 3, the artefact's desired functionality and its architecture is determined [41]. Based on this, the actual artefact is created [41].

The design of context-aware systems should follow the same steps. However, the additional connection to the environment that is necessary for developing context-aware systems should be made in activity 2 and 3, because for context-aware systems the functionality and architecture partially depends on what elements of the environment are parts of the relevant context. For functionality, insight into context is needed to determine what the system needs to be able to sense, what adaptations it needs to be able to make and what situations should lead to what adaptations. The architecture should have the necessary sensors and adaptors to be able to offer this functionality.

2.3. Representing context information and rules for adapting to context

An in-depth overview of existing work on representing context and reasoning with context information is not within the scope of this research. This area is very extensive and the literature already contains several overviews (e.g. [42–45]). The main difference between our work and the existing work lies in the way the representations and rules for reasoning are established. In our case, these follow directly from an investigation into the context of a context-aware system.

To explain the relationship with the existing works on representation and reasoning, we start with Winograd [46], who states the following: *"The hard part of this design will be the conceptual structure, not the encoding. Once we understand what needs to be encoded, it is relatively straightforward to put it into data structures, data bases, etc."* [46, p. 417]. Winograd [46] stresses the importance of having a conceptual model of context. This requires insight into the nature of context. However, there is currently no shared understanding of context [32]. This lack of shared understanding is associated with imprecise definitions of context in literature and a lack of consensus on context definitions.

In this work, we provide a definition of context that is intended to be applicable in a variety of application domains. The proposed definition is based on definitions of several other concepts, such as 'focus', 'situations', 'context elements' and 'context relationships'. The method helps designers to build the conceptual model for the context of their system based on these definitions. In fact, this is the first step of the method (see section 5.1). The conceptual model is then used as a basis for expressing rules in the system and determining its required sensors and adaptors (see sections 5.2 and 5.3). The conceptual model thus bridges the gap between investigating context and representing context in a context-aware system. This allows for a direct translation of insight into context into the rules with which the system should reason.

Winograd [46] notes that encoding or representing context itself is not the hard part. Yet, Pertunnen, Riekki and Lassila [42] note that conceptual models have received little attention in literature, compared to context representation. There is even less work on determining what context and rules should be represented in a specific context-aware system. The research that does focus on this issue covers only a specific domain, such as m-commerce [47] or web engineering [48]. This work is thus of limited application in other domains.

The work of Crowley [49] focuses on a specific domain as well, viz. observing human behaviour. However, he mentions some ideas that we will adapt and extend. Crowley [49] states that designers should only include entities and relationships that are relevant to a system task to prevent the system from becoming very complex. The relevant entities and relationships are selected by *"first specifying the system output state, then for each state specifying the situations, and for each situation specifying the entities and relations"* [49, p. 112]. The relations that he refers to are the properties of the entities. They are closer to what we define as context elements than to the context relationships that describe the impact of context in our method (see section 4.4). Determining the context relationships would be akin to determining what situation should be specified for an output state in the work of Crowley [49]. In contrast to our work, Crowley [49] does not provide explicit guidance on how to investigate this and on how to express context relationships as rules with which the system can reason.

2.4. Definitions of context in literature

The large volume of literature on context-aware systems contains many different definitions. There is currently no consensus on the definition of context in the literature [32]. The earlier work on context awareness contains definitions that use synonyms for context (e.g. situation, environment) or use examples (e.g. location) [7]. This leads to generality in the former case and to incompleteness in the latter case [50]. For designers of context-aware systems, such definitions thus respectively provide too little guidance for investigating context, or could exclude parts of context that should be included in the design of the system.

In the literature, several attempts have been made to define context for operational use without relying on synonyms or examples. Especially the work of Dey and Abowd is often used as a basis for application-specific or domain-specific definitions (see e.g. [47,51–54]). Dey and Abowd [7, p. 3] define context as follows: *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."*

According to the definition given by Dey and Abowd [7], the most important characteristics for belonging to context are 1) characterising a situation and 2) being considered relevant. However, the definition cannot be used as a basis for, for instance, quickly deciding whether something belongs to context. Their definition still leaves implicit what it means to be considered relevant to an interaction and to characterise a situation. We need to know what the notions 'relevance' and 'characterising' mean to be able to decide whether something belongs to context.

Winograd [46] argues that the definition given by Dey and Abowd [7] is too broad. He states that: *"Something is context because of the way it is used in interpretation, not due to its inherent properties"* [46, p. 405]. Zimmerman et al. [50] also mention this issue with the definition given by Dey and Abowd [7]. Their solution is to categorise context into the fundamental categories of individuality, activity, location, time and relations. According to them, the activity predominantly adds the relevance of context elements.

According to Winograd [46] and Zimmerman et al. [50], something is context because of its relationship to something else. This conforms with the interactional view on context described by Dourish [55]. According to Dourish [55, p. 5], when viewed as an interactional problem, *"contextuality is a relational property that holds between objects or activities"*. The interac-

tional view implies that something belongs to context when it has a relational property with something else. However, we still have no certainty about when exactly this is the case.

Brézillon [56] states that context cannot be spoken about out of its context. Context thus is always a context of something. According to Brézillon [56], this 'something' is a focus of an actor. Brézillon [56, p. 57] explains focus as follows: *"Context surrounds a focus (e.g. the task at hand or the interaction) and gives meaning to items related to the focus. The context guides the focus of attention, i.e. the subset of common ground that is pertinent to the current task."* He views context as knowledge and the focus helps to discriminate irrelevant external knowledge from relevant contextual knowledge. However, as Brézillon [56, p. 57] himself states, *"the frontier between external and contextual knowledge is porous"*. When in his model for task accomplishment, a discrepancy is found between the model and what a user does, the user is simply asked for an explanation [56]. The new knowledge is then added to the model. This means that Brézillon [56] does not make explicit what belongs to context either. This decision is ultimately left to the user.

The notion of a contextual element is central to the definition of context given by Vieira et al. [57], namely that a contextual element is *"any piece of data or information that can be used to characterize an entity in an application domain"* [57]. In a similar vein as in the work of Brézillon [56], contextual elements are relevant to a focus, which is determined by a task and an agent [57].

A contextual element is an attribute of a contextual entity [57], which is an entity that should be considered for the purpose of context manipulation [57]. Contextual elements can be identified from the attributes and relationships the entity has [57]. Vieira et al. [57] already note that the criterion for identifying a property as a contextual element in their case is subjective and depends on the context requirements and a conceptual model. Therefore, the question of what belongs to context becomes a question of what should be in the conceptual model. The problem of determining what belongs to context has thus been moved rather than solved.

In the work above, there is a focus on the relevance of something as arising from an activity or actor. Similarly, other work discussing relevance focuses on determining the relevance of something at runtime and dynamically defining context for the specific task or activity at hand (see e.g. [57,58]). It is important to make clear the distinction between such work and our work. The work presented in this paper is concerned with supporting the determination of what is relevant and what should be included in the context when designing a system. We thus focus on what a context-aware system should be able to sense and what adaptations it should be able to make, and in what possible situations.

To summarise, our work adds to the existing work a definition of context that can be used to clearly decide what does and does not belong to context. Furthermore, it provides a method for getting insight into context and deciding what belongs to the relevant context, based on this definition. It also provides a way to identify the sensors and adaptors that the context-aware system needs to fulfil its functional requirements. In addition, it supports the identification of the rules according to which the system should adapt, as well as expressing these rules in such a way that the system can use it to reason with.

3. Design science research method

The new method was developed using a design science approach. Design science research starts with a problem that is important as well as relevant [59]. We established the need for a new method in a triangular fashion. First, we identified the need for the method in practice when developing a B2G information sharing system in the domain of international container shipping (see section 1). Next, we searched the relevant literature for a suitable method that we could use. However, the relevant literature did not provide such a method (see section 2). Furthermore, we found that a lack of a method for investigating context is mentioned as an issue in the literature as well [32].

Design science research involves two processes, namely build and evaluate [60]. The method we propose requires a specified definition of context. Without such specification, it is not possible to clearly distinguish elements in the environment that are relevant and thus part of the context, from those that are not. We therefore started the 'build' process by searching the literature for a suitable definition of context. However, the literature did not contain such a definition (see section 2). Therefore, we developed a new definition of context (see section 4). Furthermore we formalised this definition to ensure that it is precise enough for our purposes. We then developed the method based on this new definition (see section 5).

The evaluation of a method like this is highly complex. We evaluated the method by applying it in practice when developing a context-aware system for B2G information sharing in international container-shipping. We established in the first step that a new method is needed, because the design process of context-aware systems in complex environments otherwise is likely to become either ineffective or inefficient. According to Pries-Heje et al. [61] the evaluation of a process, such as a method, can rely on the idea that a good process will lead to a good product. We can establish whether using the method leads to an effective design process by determining whether the model of the relevant context established using it is correct. We did this by letting experts in the domain check the model. We keep track of the efficiency and discussed this with the project partners. In addition, we explained the concepts of context elements and context relationships to the parties we interacted with in the project, such as project partners and interviewees and used them to discuss context. We discuss the results of the evaluation in section 6.

4. A definition of context

In this section, we provide a definition of context that can be used to develop a criterion for deciding what belongs to context in step 1 of our method (section 5.1). It can also be used to structure knowledge on context in such a way that the sensors and adaptors the system needs and according to what rules it should adapt can be easily determined (sections 5.2 and 5.3).

In section 4.1, we discuss our choice for using a logic programming paradigm to formalise the definitions. We make a distinction between semantics and syntax. We provide our basic syntax in section 4.2. Next, in section 4.3, we define some basic notions that our definition of context is built upon. In section 4.4, we provide our definition of context.

4.1. The need for a formalisation of the definitions

There are several important benefits of representing the definition of context and related concepts using a formal notation. We set out to define context as precisely as possible. As we discuss in the previous sections, this can help with clearly identifying what belongs to context and this can improve efficiency and effectivity of the design process. The process of formalising enforces a certain level of precision.

The other reasons for formalising the definitions have to do with the way in which we intent to use the definitions in the method itself. In step 1.1 of the method, the designer builds a model of the context. However, the environment that they are investigating can be highly complex. Consistently and systematically expressing knowledge about context might help designers to deal with this high complexity. More specifically, it could help them with detecting inconsistencies and gaps in knowledge. The model of the designers is based on the definition of context. The second reason for formalising the definitions is thus that it provides designers with the language and semantics to consistently and systematically model the context of their system.

In step 2 of the method, the sensors that the system needs to collect context information and the adaptors that the system needs to adapt are directly derived from the model of context. Furthermore, in step 3, the context rules for determining what adaptations to make based on the context information are also directly derived from the model of context. The formalisation supports this by providing a way to model context that allows for such a direct derivation.

In addition, the same formalisation is used to provide a way to express context information gathered by the sensors of the system at runtime. Furthermore, it allows for expressing a command to an adaptor to perform a certain action. In addition, it allows for expressing context rules. The context rules express how the system should adapt, based on the context information obtained by the sensors. This is where logic programming more specifically plays a role. By using the logic programming paradigm for our formalisation, the model of the context made by the designer can almost directly be translated to context rules. Furthermore, these rules are then suitable for use in a logic program that can be used by a reasoning component of a context-aware system to decide what is the appropriate adaptation in different situations.

This makes the step of going from a model of context to a logic program that can be used by the reasoning component of the context-aware system very small. This makes it easier for the designer to make such a translation. Furthermore, it helps to ensure that the logic program is very close to what it should be according to the model of context.

4.2. Syntax

In early work on logic programming, Kowalski [62] noted the usefulness of predicate logic in programming. In 1977, Warren, Pereira and Pereira [63] implemented a more efficient version of the logic programming language Prolog, based on this work and the work of Colmerauer [64] and van Emden [65]. Subsequently, many others build on this fundament to further extend and refine logic programming. Lifschitz [66] provides an extensive survey of the foundations of logic programming with classic negation and negation as failure. Unless otherwise stated, we follow the notation and terminology of Lifschitz [66] to describe our syntax.

We start with a nonempty set of atoms A. The choice of A depends on the language used. In our case, the atoms are simple predicates as defined below. We directly introduce variables in the language and introduce the notion of schematic atoms.

Definition 1. Given a set of constant symbols $C = \{a,\ b,\ c,\ \ldots\}$ and a set of variable symbols $V = \{X, Y, Z, \ldots\}$,

- any constant $c \in C$ is a term, and
- any variable $X \in V$ is a term.

Given a set of predicate symbols $S = \{p, q, r, \ldots\}$, an expression $p\,(t_1, \ldots, t_n)$ where $p \in S$ is an n-ary predicate symbol and $t_1, \ldots, t_n$ are terms, is an atom. If one or more terms of $t_1, \ldots, t_n$ are variables, then $p\,(t_1, \ldots, t_n)$ is called a schematic atom. If terms $t_1, \ldots, t_n$ are all constants, then $p\,(t_1, \ldots, t_n)$ is called a ground atom.

Atoms are called positive literals. Atoms preceded by a classical negation symbol '¬', are called negative literals. Following Lifschitz [66] again, we refer to a positive literal or a negative literal as a literal. A schematic atom is called a schematic

literal. A ground atom is called a ground literal. We follow the convention that terms with a capital as their first letter denote variables.

Example 1.

- $isUser\,(User)$ is an atom and a positive schematic literal
- $\neg isUser\,(mary)$ is a negative ground literal

Context rules are the same as basic rules defined by Lifschitz [66]. Again, we also introduce the notion of schematic context rule in the same definition.

Definition 2. A context rule is an ordered pair $Head \leftarrow Body$, where $Head$ is a literal and $Body$ is a finite set of literals. A context rule with head L_0 and body $\{L_1, \ldots, L_n\}$ can be written as $L_0 \leftarrow L_1, \ldots, L_n$. If the body is empty, then $\leftarrow$ can be dropped. If one or more literals of $L_0, \ldots, L_n$ are schematic literals, then $L_0 \leftarrow L_1, \ldots, L_n$ is called a schematic context rule. If $L_0, \ldots, L_n$ are all ground literals, then the rule $L_0 \leftarrow L_1, \ldots, L_n$ is called a ground context rule.

Example 2.

- The following schematic context rule expresses that when the sight that is the subject of information and a user are less than 150 metres from each other, the information should be provided to the user by the system.

$$
\begin{aligned}
&isProvidedTo\,(I, U) \leftarrow \\
&isUser\,(U)\,, \\
&subjectOf\,(S, I)\,, \\
&hasLocation\,(U, L_1)\,, \\
&hasLocation\,(S, L_2)\,, \\
&distance\,(L_1, L_2, Distance)\,, \\
&Distance < 150
\end{aligned}
$$

- The following ground context rule expresses that when Mary is at the location with coordinate 29°58′45.03″N 31°08′03.69″E, then $info$ should be provided to Mary by the system.

$$
\begin{aligned}
&isProvidedTo\,(info, mary) \leftarrow \\
&isUser\,(mary)\,, \\
&hasLocation\,\big(mary, 29°58'45.03''\text{N}\ 31°08'03.69''\text{E}\big)
\end{aligned}
$$

In addition to context rules, we need to express context relationships. The definition of a context relationship rule is similar to that of a context rule and it is also based on the definition of a basic rule by Lifschitz [66]. However, it uses a different operator to connect the head and the body of the rule.

The use of a different operator signifies that context rules and context relationship rules are different types of rules. Context rules are used to express what adaptations should be made in different situations. The body of the rule expresses the situation in which the adaptation needs to be made. The head expresses what adaptation should be made by the system, namely an adaptation that makes the head true. Context relationship rules express a dependency of the truth of the head of the rule on the situation expressed in its body. In other words, context relationship rules express that their head is true when their body is true. This is a different type of relationship. Context rules are further discussed in section 5.3. Context relationships are further discussed in section 4.4.

Definition 3. A context relationship rule is an ordered pair $Head \Leftarrow Body$, where $Head$ is a literal and $Body$ is a finite set of literals. A context relationship with head L_0 and non-empty body $\{L_1, \ldots, L_n\}$ can be written as $L_0 \Leftarrow L_1, \ldots, L_n$. If one or more literals in $L_0, \ldots, L_n$ are schematic literals, then $L_0 \Leftarrow L_1, \ldots, L_n$ is called a schematic context relationship rule. If $L_0, \ldots, L_n$ are all ground literals, then the rule $L_0 \Leftarrow L_1, \ldots, L_n$ is called a ground context relationship rule.

Example 3.

- The following schematic context relationship rule expresses that when the sight that is the subject of information and a user are less than 150 metres from each other and the information is provided to the user, then the information provided to the user is relevant.

$$
\begin{aligned}
&hasRelevance\,(I, U, relevant) \Leftarrow \\
&isProvidedTo\,(I, U)\,, \\
&isUser\,(U)\,, \\
&hasLocation\,(U, L_1)\,, \\
&hasLocation\,(S, L_2)\,, \\
&distance\,(L_1, L_2, Distance)\,, \\
&Distance < 150, \\
&subjectOf\,(S, I)
\end{aligned}
$$

- The following ground context relationship rule expresses that when the Pyramid of Cheops is the subject of information $info$ and Mary is at a location with coordinate 29°58′45.03″N 31°08′03.69″E and $info$ is provided to Mary, then information $info$ provided to Mary is relevant.

$$
\begin{aligned}
&hasRelevance\,(info, mary, relevant) \Leftarrow \\
&isProvidedTo\,(info, mary)\,, \\
&isUser\,(mary)\,, \\
&hasLocation\,\big(mary, 29°58'45.03''\text{N}\ 31°08'03.69''\text{E}\big)\,, \\
&subjectOf\,(pyramidOfCheops, info)
\end{aligned}
$$

According to Lifschitz [66], $Ground\,(R)$ stands for all the ground instances of a schematic rule R. The same function can be applied to the literals, context rules and context relationships in our case to make them stand for the set of their ground instances. For example, $isUser\,(mary)$ can be a ground instance of $isUser\,(User)$.

A program is a set of rules. In this case, we will use the context rules to reason with in a logic program to derive what adaptations need to be made based on the context information sensed. As we will further explain in section 4.4, context relationship rules have a different function and we do not need to use them in a logic program.

Definition 4. A logic program is a set of context rules.

Example 4. The following is a logic program. Its meaning is discussed in detail in section 5.3.

$$
\begin{aligned}
&\{isProvidedTo\,(I, U) \leftarrow isUser\,(U)\,, subjectOf\,(S, I)\,,\ hasLocation(U, L_1\,), \\
&hasLocation(S, L_2\,), distance\,(L_1, L_2, Dist)\,, Dist < 150, \\
&isUser\,(mary)\,, \\
&hasLocation\,\big(mary, 29°58'45.03''\text{N}\ 31°08'03.69''\text{E}\big)\,, \\
&hasLocation\,\big(pyramidOfCheops, 29°58'27.00\text{N}\ 31°08'2.21\text{E}\big)\,, \\
&subjectOf\,(pyramidOfCheops, info)\}
\end{aligned}
$$

4.3. Environment elements, situations and the focus of a context

Context-aware systems should sense context and adapt to context. To sense, there needs to be something in the real world that can be observed. For instance, the GPS coordinates of a user can be observed. Furthermore, to adapt, there needs to be something in the real world that can be manipulated; for instance, the information provided to a user can be manipulated. These things should be part of the context. First, we thus have to define what elements of the environment can be observed and manipulated by a system. These elements are candidate for being part of the relevant context of the context-aware system.

In our previous work on defining context [67], we focused on which attributes of objects do belong to context and which do not. The attributes were viewed as possibly having different values. For instance, according to this work, the attribute 'colour' of an object 'apple' could have values such as 'green' or 'red'. Relationships were also reduced to attributes, as the added value of including them was unclear at the time and attributes seemed easier to deal with. Furthermore, we built on other work that also describes context as attributes [55,57].

As the research progressed, we realised that the definition of context can be used not only for deciding what belongs to the context of a system, but also as a basis for guiding the information gathering process for getting insight into context. While investigating the context of the B2G information sharing system, we came across complex relationships that should be taken into account in the context-aware system. Using a definition that only includes attributes requires each of these relationships to be reduced to an attribute. When relationships are complicated, this is counterintuitive and makes the investigation of context more complex.

We illustrate the counterintuitivity of reducing relationships to attributes with an example. For the B2G information sharing system, we want to ensure that it supports flows of information in which businesses are willing to participate. In some cases, business A might not want business B to have certain data elements, because A and B are competitors. When data is shared in such a way that B can access it, A might not be willing to participate. However, business A might want to

share other data elements with business B that are not sensitive. Sensitivity in that case can most intuitively be described as a relationship between businesses A and B and a data element.

Attributes can be easily described as relationships. For instance, an apple can have a 'has colour' relationship with 'green' or 'red'. For the context-aware tour guide, we want to provide information about sights that is relevant to the user. What information is relevant probably depends on where the user is. In the previous work, the user would have had an attribute 'location' that has a coordinate as a value. In the new definition of context, the user has a 'has location' relationship with a coordinate.

In addition, it only makes sense to try to sense or manipulate something that could change or vary. For example, the location of a user can vary as they move around. When multiple people use a system, then who is the user might vary as well. Both of these things might be useful to sense. However, other things are less likely to vary, or are not variable at all. For example, the speed of light is not variable. For the tour guide, it might turn out that (almost) all users have a preference for being provided the information in the same language. In that case, it is not useful to sense what language users prefer.

The relationships are the elements of the environment for which we want to decide whether they are part of the relevant context. Therefore, we will refer to such a relationship as an environment element. It is important to note, that by 'environment' here we refer to the environment of the system. We made a selection of what in the environment could be manipulated or sensed. We have not yet made a selection of what of those things are part of the relevant context that should be manipulated or sensed. In this case, 'environment' thus should be interpreted in the broadest sense possible.

Informally, an environment element is a relationship between objects in the environment of a system. The objects in the environment could include physical objects, but also other things, such as qualities, or locations. Syntactically, they are represented by literals.

For example, the literal $hasLocation\,(mary, 29°58'45.03''\text{N}\ 31°08'03.69''\text{E})$ expresses that a user, Mary is at the location with coordinate 29°58′45.03″N 31°08′03.69″E. As another example, business A can have a 'willingness to participate' relationship with a flow of information and a level of willingness 'willing'. This environment element is expressed as the literal $willingnessToParticipate\,(businessA, informationFlow, willing)$.

The semantics of environment elements, i.e., whether the literal is true or not, will be determined in the system using sensors, or will be manipulated by an adaptor. In our modelling, we only want to include literals that can have different truth values.

Definition 5. An environment element is expressed by a ground literal $p(s_1, \ldots, s_n)$, where p is predicate symbol and $s_1, \ldots, s_n$ denote environment objects. A schematic literal L can be used to express the set of environment elements expressed by the literals in $Ground\,(L)$.

A state of the world, or a situation, is different from another state of the world or situation when the truth value of at least one environment element changes. For a system to be context-aware, it should sense or adapt to these differences when they are relevant. Furthermore, it should only consider situations that could exist in the real world, and for instance not situations that are inconsistent.

A situation is a state of the world determined by the environment elements that are true. Syntactically, they are represented by a set of ground literals.

For example, there could be a situation in which Mary has a 'has location' relationship with coordinate 29°58′45.03″N 31°08′03.69″E and in which she is a user of the context-aware tour guide. This can be described by set $\{hasLocation(mary,$ 29°58′45.03″N 31°08′03.69″E$), isUser\,(mary)\}$.

This situation is different from one in which the location of Mary is 29°58′31″N 31°08′16″E. This situation can be described by set $\{hasLocation\,(mary,\ 29°58'31''\text{N}\ 31°08'16''\text{E}),\ isUser\,(mary)\}$. In addition, the set of literals $\left\{\begin{array}{l} hasLocation\,(mary,\ 29°58'31''\text{N}\ 31°08'16''\text{E}), \\ \neg hasLocation\,(mary,\ 29°58'31''\text{N}\ 31°08'16''\text{E}) \end{array}\right\}$ does not express a situation as it is inconsistent.

As another example, for the B2G information sharing system, a situation could exist in which business A has a sensitivity relationship with data element 'client name' and business B. This situation is different from one where the client name is not sensitive from B according to A. Respectively, these situations are expressed by the sets $\{isSensitive\,(businessA, clientName, businessB)\}$ and $\{\neg isSensitive\,(businessA, clientName, businessB)\}$.

Definition 6. A situation is expressed as a nonempty and consistent set of ground literals $\{L_1,\ L_2, \ldots\}$, where each L_i expresses an environment element that is true. A finite set $\{L_1, \ldots, L_n\}$ describes part of a situation and stands for all situations in which $L_1, \ldots, L_n$ are true.

The origin of the notion of context lies in the domain of linguistics [68]. In this domain, the meaning of a text has to be constructed based on the surrounding text [46]. This surrounding text can be viewed as the context of the text for which the meaning is constructed.

Outside the domain of linguistics, context is always a context of something as well. We need to identify what this something is and what to call it. In linguistics, this 'something' (i.e. the text for which the meaning is constructed) is called a 'focal event' [69]. In the domain of context-aware systems, for Dey [70] it is the interaction between a user and

an application, and for Brézillon [56] and Vieira et al. [57] it is a focus. Accordingly, we refer to the thing that context is a context of as a focus. Context is thus the context of a focus.

Now we have a name for it, we need to determine what a focus is in the case of context-aware systems. We want our definition of context to be generic enough to make our method useful for a variety of application domains in which a context-aware system is designed. Therefore, we believe that limiting the focus to the interaction between a user and an application, like Dey [70] does, is too restrictive. We have to look more broadly at and examine what the nature is of the relationship between contexts and their focus for context-aware systems.

The designer of a context-aware system has a goal and they want to design the system to reach that goal [33,71]. This design goal is determined by the designer based on the design goals of the different stakeholders in the context-aware system. The design goal is the same in all situations [33]. However, whether the design goal is reached can depend on the situation. Everything that could affect whether the design goal is reached at runtime should belong to the context; that is, the situations in which the design goal is not reached should be sensed. This should then result in an adaptation by the system that changes the situation in such a way that the design goal is reached. Anything that does not impact whether the design goal is reached should not belong to the context that the designer should take into account. The focus of the context should thus be related to the design goal of the designer.

The focus of context is the environment element that a designer of context-aware systems needs to be true to reach their design goal. As it is an environment element, it is syntactically represented by a literal, in the same way as other environment elements.

For example, a designer can have the design goal of developing a context-aware tour guide that provides information about sights that is relevant to users. This goal is reached when the information provided is relevant to users. This focus can be expressed by the literal $hasRelevance(ProvidedInfo, User, relevant)$ when there are several different levels of relevance. It can be represented by the literal $isRelevant(ProvidedInfo, User)$, when the information is viewed as either relevant or not relevant.

Note that what information is relevant is the part that is important here. This is what should be adapted to in the end. This is not the case for what information is provided to the user. Just providing information without requiring relevance of the information does not require context awareness. Therefore, the predicate we use is *hasRelevance* and not $isProvidedTo$. As discussed in section 5.2, the decision for taking into account context for reaching a goal is in part one that should be made by the designer on beforehand. As the focus is directly dependent on the goal of the designer, the designer has as much freedom to select a focus as they have to select a goal.

In the case of the B2G information sharing system, a designer can have the design goal of developing a context-aware system that supports flows of information in which businesses are willing to participate. This goal is reached when the system supports flows of information in which businesses are willing to participate. The focus can be expressed by the literal $willingnessToParticipate(Business, SupportedInformationFlow, willing)$.

It is important to note the use of schematic literals in the examples here. The design goals of a designer are usually high level. The designer's goal is not to provide Mary and Bob with relevant information, but to provide all users with relevant information. Schematic literals can be used to reflect this.

Definition 7. A focus of context can be expressed using a literal in the same way as other environment elements.

4.4. Context relationships, context elements and context

At first sight, achieving a definition of context seems problematic, since what belongs to context might be different for each context-aware system. However, context is determined by its relationship with its focus. In fact, something is only context if it has some *context relationship* with the focus. For instance, the weather forecast only belongs to the context of the tour guide if it has a context relationship with the relevance of information provided to the user. The type of relationship is not specific to a certain focus, but the same for all foci. In this way, it can be used to formulate a definition of context from which what belongs to the context of a specific focus can be derived.

A context relationship is a relationship between a focus and a set of environment elements, where in each situation where these environment elements have the same truth value, the focus has the same truth value. We say that these situations restrict the focus. Syntactically, context relationships are represented by context relationship rules.

Note that the context relationship is not the same type of relationship as the environment elements. It connects different environment elements with each other. The context relationship is thus on a higher level and can more naturally be represented by an operator than by a predicate.

For the context-aware tour guide, the focus is the relevance of the information provided to the user. Let us assume that information about a sight is relevant when the user is close to the sight and the information is about the sight. This means that in all situations in which the 'has location' relationship of the user has a coordinate within a few metres of a sight and the 'subject of' relationship of the data provided is this sight, the value of the focus is such that the information provided is relevant to the user. In that case, there is a context relationship. For completeness, it should be noted that, for instance, the 'is user' relationship between the user and the context-aware system and the 'provided to' relationship restrict the focus as well.

For the context-aware B2G information sharing system, the focus is the relationship of willingness to participate between a flow of information, a business and a level of willingness. We found that businesses are not willing to participate in a flow of information when the data is sensitive to them and there is a system in the flow of information that broadcasts it. Therefore, the previously mentioned sensitivity relationship has a context relationship with the focus. The same is the case with the broadcasting relationship a system has with information and the 'being part of' relationship of a system and a flow of information. This context relationship can be expressed as follows.

$$\begin{aligned}&\neg willingnessToParticipate\,(Business, SupportedInformationFlow, willing) \Leftarrow\\ &sensitivite(Business, Competitor, Data),\\ &partOfFlow\,(SupportedInformationFlow, System)\,,\\ &broadcasts(System, Data)\end{aligned}$$

It is important to note that businesses might be either willing or unwilling to share when the information is not sensitive, the system is not part of the flow of information or the system does not broadcast the information. This is possible considering the context relationship we have identified. There is only a dependence on the focus when all the environment elements have the values mentioned in the example. Context relationships, however, might in other cases constrain the value of a focus for another truth value of their context elements as well. In addition, there might be multiple sets of context elements that have a context relationship with a single focus.

For example, what the authors of this paper have for dinner does not have a context relationship with either the focus of the tour guide or the focus of the B2G information sharing architecture, as this does not restrict them.

Definition 8. A context relationship can be represented by a ground context relationship rule $L_0 \Leftarrow L_1, \ldots, L_n$, where L is a literal representing the focus and where $L_1, \ldots, L_n$ are literals representing the context elements that restrict the focus. A schematic context relationship rule R can be used to represent the set of context relationships represented by $Ground\,(R)$.

It is important to note the effect of using schematic rules to represent context relationships. For example, the distance between two locations will usually not be subject to change. This means that the 'distance' relationship between locations will not be part of the context relationship in the case of the context-aware tour guide. This means that in theory, expressing that the location of the user and that of the sight have to be within a certain distance could be done by listing the same rule over and over again, each time with different locations that are within this distance. In practice, this would be impossible. It would be much easier to just use a schematic rule that stands for this set of rules and includes the relationship of 'distance' to constrain the instances that are represented by the schematic rule. There is no obvious prohibition against allowing this, as long as it is clear that from a semantic point of view, these literals represent constraints and are not part of the set of environment elements that have a context relationship with the focus. In the case of the context-aware tour guide, such a schematic rule, including the constraints could be expressed as follows (constraints are underlined):

$$\begin{aligned}&hasRelevance\,(ProvidedInfo, User,\ relevant) \Leftarrow\\ &isUser\,(User)\,,\\ &hasLocation\,(User, Location1)\,,\\ &providedTo\,(User, ProvidedInfo)\,,\\ &\underline{hasLocation\left(Sight, Location2\right)},\\ &\underline{distance\left(Location1,\ Location2,\ Distance\right)},\\ &\underline{Distance < 10,}\\ &\underline{subject\left(ProvidedInfo, Sight\right)}\end{aligned}$$

Using the notion of context relationship, we can determine whether an environment element belongs to context. A context element of a focus is an environment element that is part of a set of environment elements that have a context relationship with the focus. As it is an environment element, it is syntactically represented by a literal in the same way as other environment elements.

For example, the location of a user is a context element of the focus in the tour guide example. If it is unlikely that the location of a sight will change, or if it is impossible for it to change (e.g. in the case of the Pyramid of Cheops), then the 'has location' relationship of the sight with a coordinate does not have a context relationship with the focus, because its truth value never changes and therefore it is not an environment element.

Furthermore, the sensitivity relationship is a context element of the focus of the B2G information sharing example, as it has a context relationship with that focus. In contrast, what the authors of this paper have for dinner is not a context element, as it does not have a context relationship with the focus.

Definition 9. A context element can be expressed by a literal in the same way as other environment elements.

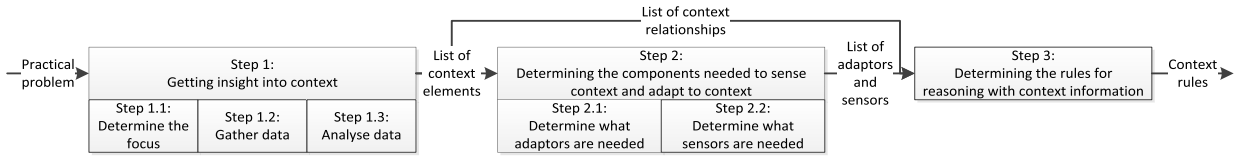

Fig. 1. Overview of the steps in the method and their input and output.

When an environment element is a context element of a focus, this means that it is relevant to the designer. A designer achieves their design goal when the focus has a certain value. To achieve the design goal, they thus have to design the context-aware system such that the focus has this value when it is used. A context element of the focus influences the value of that focus. Therefore, the system needs to be designed such that it can sense the context elements and manipulate them if the focus has an undesired value. This makes the context element relevant to the design and therefore to the designer.

The definition of context is based on the other notions defined above. The context of a focus is the set of all its context elements.

For example, the context of the focus of the tour guide example includes the location of a user. In addition, the context of the focus of the B2G information sharing example includes the sensitivity relationship. Syntactically, context it is represented by a set of literals.

Definition 10. The context of a focus can be expressed by a set of literals $\{L_1, L_2, \ldots\}$, where each L_i expresses an environment element.

5. A method for designing context-aware systems

In this section, we present the three steps of the proposed method for designing context-aware systems based on insight into context, viz. 1) getting insight into context, 2) determining the components needed to sense context and adapt to context, and 3) determining the rules for reasoning with context information. The method takes a practical problem that a designer wants to solve as a starting point. In step 1, it is determined what context relationships and context elements the system should take into account to solve the problem and attain the designer's goal. In step 2, the list of context elements is used to determine what sensors and adaptors are needed. In step 3, the list of sensors and adaptors, together with a list of context relationships from step 1, is used to derive and express the rules according to which the system needs to adapt. Steps 1 and 2 consist of several sub-steps. For each sub-step we provide an illustration of how it can be performed for the examples of the tour guide and the B2G information sharing system. Fig. 1 provides an overview of the steps in the method

5.1. Step 1: getting insight into context

The objective of the designer in the first step is to get insight into context. The first step of our method should be preceded by the identification of the practical problem that the context-aware system should solve and a determination of the relevance of that problem. How to do this is already described extensively in scientific literature (see e.g. [40,41]). This is thus not new, and therefore not part of our method.

However, what context should be taken into account in the design of the system is related to the design goal of the designer. We can derive this design goal from the problem that is identified. The specification of the practical problem is thus the input for this step.

The sensors and adaptors of a context-aware system and the rules that the system should reason with depend on the context that should be taken into account (see section 1). This means that in this step insight into context needs to be gained to determine what belongs to context and what the impact of different situations is. Information about this should thus be the output of steps 2 and 3. This information should be structured in such a way that the necessary sensors and adapters for the architecture, as well as the rules, can easily be derived from it.

For the first step of our method, we propose that the designer performs the following sub-steps: 1.1) determine the focus, 1.2) gather data, 1.3) analyse the data.

5.1.1. Step 1.1: determine the focus

The overall process for determining the focus (step 1.1) is shown in Fig. 2. The input of this step is the practical problem and the output is a focus.

The focus of the context is related to the design goal of the designer. The design goal of the designer, in its turn, is based on the problem that they want to solve with their system. To perform step 1.1, the designer can use the specification of their problem to specify their design goal.

The design goal of the designer can be viewed as solving the practical problem. The design goal can thus be expressed as what the system should be able to do at a very high level in order to solve the problem. A design goal is reached when

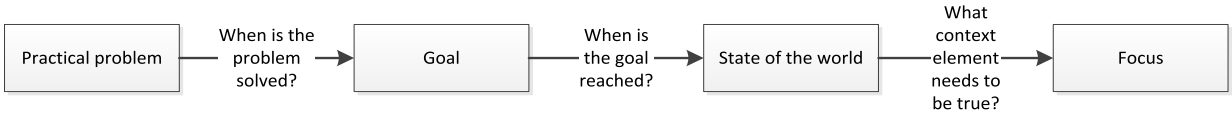

Fig. 2. Determining the focus.

the world is in a certain state. Therefore, the next step for the designer is to describe the state of the world when their design goal is reached.

According to Definition 7, a focus of a designer is the environment element that the designer needs to be true to reach their design goal. This relationship can be identified from the description of the state of the world.

Fig. 3 shows the steps for deriving the focus for the tour guide example. Fig. 4 shows the steps for deriving the focus for the tour guide example.

It is possible that a more complex design goal can lead to multiple foci, or that there are multiple design goals. This should not be a problem. However, each of the foci will have its own context relationships and context elements and will require its own investigation by the designer.

Illustration step 1.1 (tour guide):

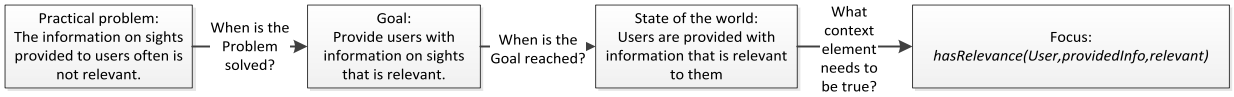

Fig. 3. Deriving the focus for the example of the context-aware tour guide.

Illustration step 1.1 (B2G information sharing system):

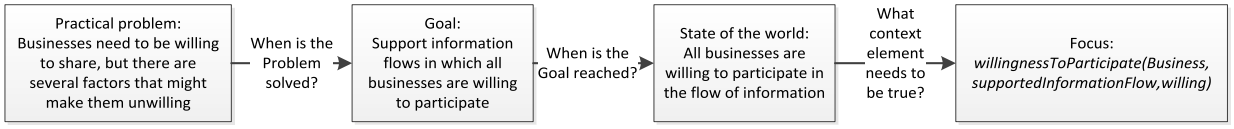

Fig. 4. Deriving a focus for the example of the B2G information sharing system.

5.1.2. Step 1.2: gather data

The overall process of performing step 1.2 is shown in Fig. 5. The input is the focus from step 1.1 and the output is a table with situations that restrict the focus.

After the designer has identified the focus, they should first select the data collection methods and sources that they will use to investigate context. An important requirement is that the data that the designer gathers should provide information on what environmental elements restrict the focus. There are many different approaches that could be taken. A designer could, for instance, perform case studies in which the focus has certain values and then determine why the focus has these values. Another possible approach is to do a literature search, using a description of the focus. Furthermore, a designer could conduct interviews and ask the interviewees directly or indirectly what they think impacts the focus. In addition, the usual considerations, such as the accuracy and accessibility of data, will also play a role in making a choice.

Illustration Step 1.2 (tour guide): For the tour guide example, the selection should depend on what data says something about the level of relevance of information about sights to users. This could very well be the users themselves, and the designer might ask them to complete questionnaires or interview them about in what cases they find information relevant. Among the other possibilities are interviewing experts or doing a literature study on the matter.

Fig. 5. Gathering data on the context of a context-aware system.

Illustration Step 1.2 (B2G information sharing system): For the willingness focus, we needed information about in what situations businesses are and are not willing to participate in information flows. We did a secondary analysis of case study data from a project in which different systems for information sharing were implemented and tested in the container shipping domain. The systems had different designs and were used by different parties. The case study data thus offered insight into the reasons for these differences and the concerns of businesses. We studied different deliverables and reports describing the design of the systems, the progress of the project and the obstacles the researchers came across. Furthermore, we studied some of the case study notes. The data provided a broad perspective on what things affect the willingness of businesses. Furthermore, this data was rich and accessible.

It is hard to identify the best data source and the best data collection method without knowing the focus that data should be collected for. However, as we are investigating what belongs to context, we can say that data gathering should be of an exploratory nature. This is most important in the early stages of the investigation to ensure that no important parts of the context are excluded. In later stages, additional support should be sought for the context elements and relationships found. Unfortunately, the exploratory approach can be in conflict with the feasibility and efficiency of the data gathering. In a large search space, such as in the case of the complex environments discussed in the introduction, it would be easy to end up investigating a lot of things that later turn out to be irrelevant to the design of the system.

We propose several strategies to minimise the time that the designer spends on investigating things that turn out not to matter for the design of the context-aware system. First of all, we already restricted the search space by using the focus to select appropriate data collection methods and data sources. Second, we can restrict the kind of information that the designer should gather. The designer only needs to know in what situations the focus is restricted. The designer will need very specific descriptions of the situation and the way in which the focus is restricted. However, they do not need to do a more in-depth analysis of why the focus is restricted in that situation, than is necessary to determine that there is a connection and that the information is reliable. For instance, the designer needs to know in what situations a user finds information about sights relevant, but does not need to know the details of theories on human information processing that make that information relevant. After all, this is not something that the context-aware system can directly take into account.

An additional way to increase the efficiency of the data gathering process is to provide a way for designers to quickly decide whether something has a context relationship with the focus. We therefore provide a criterion for deciding whether a relationship in the environment has a context relationship with the focus. Furthermore, we provide a simple test to decide whether the criterion is met for a set of environment elements. This criterion can be used to discern pieces of data that are interesting for further analysis and those that are not. The criterion follows directly from the way we defined the notion of environment elements and context relationships in section 4.

Criterion: A relationship in the environment has a context relationship with a focus if and only if:

- whether the relationship exists can vary (making it an environment element), and
- it is part of a set of environment elements, such that in each situation where these environment elements have the same truth value, the focus has the same truth value.

The criterion is met when a situation is that restricts the truth value of the focus and the relationship is part an environment element that is part of this situation. Of course, we cannot list all possible situations to check whether the criterion has been met, because in the real world there are far too many other environment elements that vary. Therefore, it is also not possible to be certain that all environment elements belonging to a set that have a context relationship with a focus have been found.

The solution is to reduce the testing of the criterion to testing whether the information collected or analysed by the designer supports the conclusion that the criterion is met. The designer should thus determine that their information supports the conclusion that the focus is restricted to a certain truth value in a specific situation. If information is found that indicates that the truth value is restricted for a focus to either true or false in a certain situation, the criterion is met for all the environment elements in the situation. To test whether the criterion is met, it is thus enough to describe the situation that impacts the focus. This is more intuitive and efficient, and at this stage we do not need to discern the different context elements.

Furthermore, depending on what data the designer gathered, they may need to generalise. Consider an example in which Mary is interviewed by the designer and she states that she thinks information about the Pyramid of Cheops is relevant when she is near the pyramid. On the basis of this, we could say that in the situation in which Mary uses the system and her location is near the Pyramid of Cheops and the information provided is about the Pyramid of Cheops, the information provided to Mary is relevant. However, unless the system is designed specifically for Mary and the pyramid, this is not very informative. Generalisation in this case is easy: replace ‘Mary’ with ‘the user’ in the description of the situation and ‘the Pyramid of Cheops’ with ‘the sight’. By generalising, the description stands for a whole set of situations with particular properties in which the designer believes that the focus is restricted.

Of course, the designer should be confident that they can make generalisations based on the information that they gathered. If a relationship meets the criterion but the designer is not sure whether they can generalise, then gathering further data on the situation that relates it to the focus and similar situations and their impact on the focus might be

Table 1
Table shell for testing whether the criterion is met.

Restriction on focus	Situation	Support
Description to what value focus is restricted	Description of the situation in which the focus is restricted	Reference to a data source or citation

fruitful. It is not always necessary for designers themselves to generalise. For instance, when scientific research shows that location and relevance are related, designers do not need to generalise.

To ensure that the designer has found the appropriate information, they should describe explicitly and precisely the restriction on the value of the focus as well as the situations. Designers should ensure that everything they say in the description of the situation follows from the information that they have. We propose that designers fill in a table shell similar to that in Table 1. If they can fill this in based on the information they have gathered, they have found environment elements that have an impact on the focus and that meet the criterion. If they cannot fill it in, then they cannot conclude that the criterion is met, based on the information they gathered. When filling in the table shell, designers should keep in mind that situations should be possible in the real world.

Illustration Step 1.2 (Tour Guide):

Table 2
Testing whether the criterion is met for the tour guide.

Restriction on focus	Situation	Support
The information provided to the user is relevant	The user is near the sight. The information provided is about the sight.	Interview 1,00h21m: Interviewee: *'I think the information about the Pyramid of Cheops is relevant to me, when I am near it.'*

Illustration Step 1.2 (B2G information sharing system): For the investigation of context for this example, we went through the case study data. For everything we thought might restrict willingness, we attempted to fill in the table shell and find additional information. We explicitly made the step to generalise from specific situations by replacing objects in our descriptions with their type (e.g. 'freight forwarder' with 'business'). When different situations generalised to a similar overall description, we added the new situation to the older one as support. Table 3 below shows an example of the results of filling in the table shell. We took an example for which the support was already generalised.

Table 3
Testing whether the criterion is met for the B2G information sharing system.

Restriction on focus	Situation	Support
Businesses are not willing to participate in the flow of information.	The data in the flow is sensitive for a business to another business. The data is shared with the business that the data is sensitive to.	Doc 1: *'A freight forwarder that books a container transport on behalf of an exporter at an ocean carrier, is not willing to share the name of this exporter, because they fear that the exporter and ocean carrier might bypass him ("disintermediation"), and make a direct container transport booking with the carrier.'*

By filling in the table shell, designers can determine what things are and what things are not interesting to further look into. Then, they can concentrate their data gathering efforts on further specifications of the context elements and relationships that they found, and on finding more support for them. In fact, to make everything computable, the description of the situation and the value for the focus need to be as specific as possible. In the tour guide example, the designer might, for instance, attempt to try to find out what exactly is near enough to the sight for the information to be relevant. The filled-in table shells are further analysed in step 1.3. This means that it is important for the designer to add further specifications to the table shell.

Filling in the table shell not only serves as a test that the criterion is met, but is also used in further steps to determine what the context relationships and context elements are of the focus. As the information in the table is further processed later on, it seems wise to include a reference to the data source or citation of the text that the information is based on. This can help to ensure reliability and allow for reinterpretation in later steps.

5.1.3. Step 1.3: analyse the data

The overall process of performing step 1.3 is shown in Fig. 6. The input is a table with descriptions of situations from step 1.2. The output is a list of context elements and context relationships.

As the output of step 1, we require the information found about the context to be structured in such a way that it is easy to identify what sensors and adaptors the system should need and such that rules can be abstracted from it. The first thing that needs to be done to achieve this is to abstract the environment elements and their truth values from the situations described in the table shell that is filled in step 1.2.

There are many ways to do this. One is by looking at the descriptions of the situations and for each one identifying all the physical things in the real world that are mentioned. Physical things are physical objects in the world, such as a user.

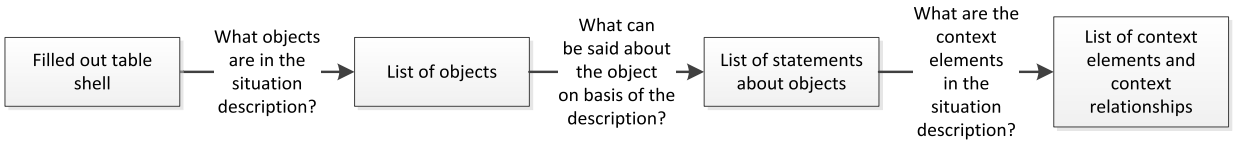

Fig. 6. Analysing the data on the context of a context-aware system.

Table 4
Table shell for recording information on context relationships.

Name	Restriction on focus	Situation	Context elements	Support
Name of the context relationship	Description to what value focus is restricted	Description of the situation in which the focus is restricted	List of environment elements identified in the description	Reference to a data source or citation

Other things are not physical. For example, qualities are things that are attributed to those objects, such as colour or speed [72], but are not physical objects themselves. Both physical things and qualities can be related in an environment. To find the environment elements, everything that is true about the physical objects in the situation can be listed and then it can be determined what of these things vary.

Illustration Step 1.3 (tour guide): On the basis of the filled-in table shell (Table 2), we can identify the following objects: the user, the location of the user, the sight, the location of the sight, and the information. Then we can list everything that is true about these objects in the situation: there is a user, there is a sight, there is information, the user has a location, the user is provided with the information, the location of the user is near the location of the sight, the information is about the sight, and the sight has a location.

Illustration Step 1.3 (B2G information sharing system): For the first situation in the table shell resulting from step 1.2 (Table 3), we can identify the following objects: the data, the flow of information, a business, and another business. What we can say about these objects is the following: there is data, there is a flow of information, there is a business, there is another business, the data is sensitive from one business to the other, and the data is shared with the other business in the flow.

In the previous section, we discussed how to express environment elements (Definition 5). We can use this here to express the environment elements that are found during the analysis. We can use ground literals when we know about or want to express a specific instance of an environment element. For instance, when the location of Mary is a specific coordinate, we express this using a literal, for example $hasLocation\,(mary, 29°58'45.03''\text{N}\ 31°08'03.69''\text{E})$.

When we want an adaptor to provide Mary with piece of information $info$, we can express this using ground literals as well, for instance $isProvidedTo\,(info, mary)$. The ground literal then expresses what the desired situation is and the adaptor should perform an action to achieve this situation, for example provide $info$ to Mary.

In the previous step, we already generalised the information we found on context. For instance, we replaced 'Mary' with 'the user'. This generalised information can be expressed using schematic literals that represent a set of instances. As discussed in section 4.4, this avoids having to provide a very long list with similar context relationships. It also introduces the need to express constrictions in some cases, which can be added as literals as well.

We can add the schematic literals expressing each of the environment elements to extend the table shell we used for testing the criterion (Table 1). In each row, we describe a situation and the way in which the focus is restricted in that situation. We are thus describing the relationship between context and the focus, or in other words, a context relationship. Therefore, we can also add a column to name the context relationship. The resulting extended table shell is shown in Table 4.

Illustration Step 1.3 (tour guide): Table 5 shows the literals that can be assigned to the statements for the tour guide example.

Illustration Step 1.3 (B2G information sharing system): Table 6 shows the recorded information on context relationships for the B2G information sharing system. The column with the support was left out to save space.

When filling in the table shell, it could turn out that information is still missing. In that case, steps 1.3 and 1.2 should be alternated until all information that needs to be recorded in the table shell has been acquired.

5.2. Step 2: determining the components needed to sense context and adapt to context

In this step, we only need to look at the environment elements that are classified as context elements, because these are the environment elements that impact the focus and thus should be sensed and, in some cases, manipulated by the

Table 5
Recording information on context relationships for the tour guide example.

Name	Restriction on focus	Situation	Context elements	Support
Proximity of the user to the sight	The information provided to the user is relevant	The user is within 150 metres of the sight and the information provided is about the sight	$isUser(U)$ $subjectOf(S, I)$ $haslocation(U, L_1)$ $haslocation(S, L_2)$ $isProvidedTo(I, U)$ $distance(L_1, L_2, \text{Dist})$ $Dist < 150$	Interview 1, 00h21m: Interviewee: *'I think the information about the Pyramid of Cheops is relevant to me when I am near it.'* Field testing: *a sight and a user are near when they are closer than 150 metres.*

Table 6
Recording information on context relationships for the example of the B2G information sharing system.

Name	Restriction on focus	Situation	Context elements
Do not share sensitive data	Businesses are not willing to participate.	The data in the flow is sensitive for a business to another business. The data is shared with the business that the data is sensitive to.	$isBusiness(BusinessA)$ $isBusiness(BusinessB)$ $isSensitiveTo(Data, BusinessA, BusinessB)$ $isSharedWith(Data, BusinessB)$

system. We can thus discern two types of context elements, namely sensor elements and adaptor elements. In this step, we should determine which is which. The output of this step is a high-level, partial description of the architecture of the context-aware system that includes only the sensors and the adaptors, and their input and output and connections to the environment.

In the literature, there are several proposals for what the overall architecture of a context-aware system should look like (see e.g. [46,73–75]). It is not up to us to decide which one of them is best or which one the designer should choose. However, any architecture that a designer considers most apposite for the design of their system can be used in our method. This is possible, as all architectures for context-aware systems have some components and connections in common. The design choices we want to help the designer with only concern these common components and connections.

The similarities between the possible architectures are dictated by the nature of context-aware systems. Ultimately, a context-aware system should sense context information and adapt to it. Its architecture should thus always include sensors, adaptors and a component for reasoning with the information from the sensors to derive what adaptors should make what adaptation. This means that the sensors and adaptors should have some direct or indirect connection to the environment and to the reasoning component.

It is exactly these sensors, adaptors and connections to the environment that are common to the architectures that we want to support making design decisions on. We want to support designers in basing their design on insight into context. What sensors and adaptors should be included in the architecture is directly determined by what context the system should take into account. Furthermore, the same is the case for what things in the environment the sensors and adaptors should directly or indirectly connect to. Therefore, we only want to guide designers in making choices on this.

The functionalities that a system offers can be divided into basic functionalities and adaptive functionalities [37]. The sensors and adaptors provide these adaptive functionalities. The architecture of the system should also include components for providing these basic functionalities. For example, for the tour guide system, the basic functionality is to provide information about sights. The architecture needs to include the components that can deliver that functionality, such as a database with information about sights and a screen to present the information. We assume that the system to provide the basic functionalities is already designed at the start of this step.

What functionalities belong to basic functionalities or to the adaptive functionalities is in part a design choice that a designer has to make up front. A designer has to make a choice on what goals they want to reach using context awareness. This choice should be based on whether they believe that reaching the goal requires providing different, or adaptive, functionality in different situations. Only after they make this choice, the proposed method plays a role. However, in case they base a focus on such a goal but they have a hard time finding context relationships for that focus, it is a sign that the way in which the goal is reached probably does not depend on context. In that case, they might go back on their choice.

Step 2 of the method can be divided into two sub-steps: 2.1) determine what adaptors are needed and 2.2) determine what sensors are needed.

5.2.1. Step 2.1: determine what adaptors are needed

The overall process of performing step 2.1 is shown in Fig. 7. The input for this step is a list of context elements. The output is a list with descriptions of adaptors that can manipulate context elements.

To complete this step, we need to identify for each context element whether it could and should be manipulated by the system. By manipulation, we mean that the system performs an action that changes the truth value of the context element. This allows the system to adapt and change the situation, such that the value of its focus corresponds to its design goal.

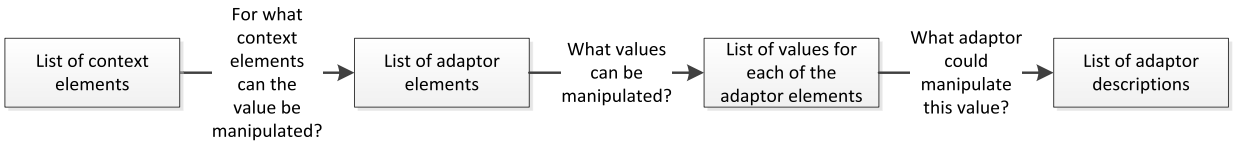

Fig. 7. Determining the adaptors needed for a context-aware system.

We call the component of the system that performs this action an adaptor. The input of the adaptor is a decision of the reasoning component on what value the adaptor needs to achieve for the context element. This value is expressed as a literal.

It is important that for each context relationship, there is at least one context element that can be manipulated by the system. Otherwise, the system cannot adapt and it is not possible for the system to take the context into account. Of course, there can be more than one context element in a context relationship that can be manipulated. However, whether we need more than one context element to be manipulated depends on the type of context relationship.

Context relationships describe in what situations a focus is restricted to a certain value (e.g. the information provided is not relevant). The design goal of the designer is to ensure that the focus has a certain value (e.g. the information provided is relevant). On basis of this, we can distinguish two types of context relationships, namely negative and positive context relationships. A positive context relationship restricts the focus to a value that conforms with the design goal of the designer. A negative context relationship restricts the focus to a value that does not conform with the design goal of the designer.

For example, the restriction on the focus such that the information provided to the user is relevant in a situation in which the user is within 150 metres of the sight and the information provided is about the sight, is a positive context relationship for the tour guide example (see Table 5). The restriction on the focus to 'not willing to participate' in a situation in which the data in the flow is sensitive for a business to another business and this data is shared with the business that the data is sensitive to, is a negative context relationship for the example of the B2G information sharing system (see Table 6).

For a positive context relationship, we need to ensure that all its environment elements have the value specified. This means that the system should contain adaptors for each context element for which this is possible. For a negative context relationship, we only need to ensure that at least one context element has a value different from that specified. This means that it is sufficient for the designer to choose one context element that should be manipulated, and that the system should contain this adaptor.

In principle, it could also be possible to manipulate multiple context elements of a negative context relationship. However, for negative context relationships, this causes these manipulations to have a disjunctive relationship with each other; either one can be performed to ensure that the focus is not negative. This offers the advantage of having multiple options for manipulation. However, it also leads to complications, as it introduces a form of nondeterminism. It thus requires a more complex reasoning mechanism to deal with this, which might not weigh against the advantages of having multiple options. Therefore, we choose one context element to manipulate for negative context relationships.

What context element to manipulate will often be an obvious choice. It is often clear what context elements cannot be manipulated, because it is not possible or it is undesirable or too costly. These options can be eliminated. The remaining context elements are then the adaptor elements.

Illustration Step 2.1 (tour guide): It would not be possible for the tour guide system to change the location of the Pyramid of Cheops. It might be undesirable to tell users to go to another location to ensure that the information they receive is relevant. Manipulating what information is provided to the user clearly is feasible and desirable for the tour guide.

Illustration Step 2.1 (B2G information sharing system): It is not possible to manipulate the sensitivity of information in the system. However, it is possible to manipulate with whom the data is shared in the information flow.

To determine what adaptors the context-aware system requires, the designer needs to first determine how the value of an adaptor element could be manipulated to achieve the value that is appropriate for the situation. For this, the designer needs to look at the terms of the environment element and see what needs to be changed, and then determine what components the system requires to perform the appropriate manipulations. As there might be several possibilities, for which it is easier or harder to find a component that can perform them, the designer might need to alternate between the selection of possible manipulations and finding accompanying components, before an appropriate component is found. Each component should be described and incorporated in the overall architecture of the context-aware system.

Illustration Step 2.1 (tour guide): In the tour guide example, we found a context element in which there is a relationship between a user and information, such that the user is provided with the information. This is represented in Table 5 as $isProvidedTo(I, U)$. The context relationship is positive, so if $isProvidedTo(I, U)$ has the value 'false', it should be manipulated such that it has the value 'true'. The system cannot influence who the user is. However, it would not be hard to make the system manipulate what information a user is provided with. To do so, a component should perform the actions of finding information and providing it to the user. We do not need a very complex component to do this in this case. Let us assume that the architecture providing the basic functionality of the tour guide includes a screen to show users information

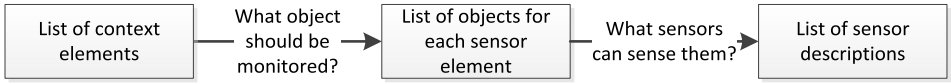

Fig. 8. Determining the sensors needed for a context-aware system.

about sights. It also includes a database with information about sights. The reasoning component will decide what information the user should be provided with and thus how to change what I is instantiated with. The required instantiation for I is provided to the adaptor. All the adaptor needs to do is search the database for the appropriate information and send this information to the screen.

Illustration Step 2.1 (B2G information sharing system): The adaptor element for this example is represented as $isSharedWith(Data, BusinessB)$ in Table 6. As the context relationship is negative, the system needs to adapt such that $\neg isSharedWith(Data, BusinessB)$. In other words, the adaptor should ensure that the data is not shared with business B. There are several ways in which the sharing of data can be prevented, for instance by blocking the sending of the data to other parties. However, in our case the B2G information sharing system shares information using a distributed ledger. This means that if data is added to the ledger, all parties get a copy. In such a design, to ensure that the data is not shared with business B, the data can be encrypted and business B is not provided with a key. To make such a manipulation possible, two components are needed, namely a component that encrypts data and a component that controls access by providing or not providing a key.

The adaptor components need to connect directly or indirectly to all objects connected in the context element to be able to manipulate it, and to the reasoning component to get the input necessary for it to know what actions to perform. Thus, in the tour guide example, it needs to connect in some way to the information and to the user. For the information, this is done by searching the database with the information about sights. For the user, this connection is less direct and is done via another component, namely the screen of the system. In the example of the B2G information sharing system, the adaptor components need to connect to the data, which is done by encrypting it, and to the businesses, which is done by the access control component.

5.2.2. Step 2.2: determine what sensors are needed

The overall process of performing step 2.2 is shown in Fig. 8. The input for this step is a list of context elements. The output is a list with descriptions of sensors that can sense the context elements.

To complete this step, we need to identify for each of the context elements that are not adaptor elements how it can be determined whether they are true in a situation. First, a decision should be made on what object in the world will need to be monitored. Then a measurement for establishing whether the object has a certain relationship should be found. Subsequently, it needs to be determined what component could carry out the measurement. Just as in the case of the adaptors, this might require some alternations between identifying the object to monitor, identifying possible measurements and finding an appropriate sensor. The last step is to determine what connections the sensor should have to the environment.

Each environment element, according to Definition 7, connects things in the environment. For a sensor to be able to monitor, it should monitor a physical object in the environment. The object that the sensor should monitor should be one of the objects in the context element. If there are multiple objects that could be monitored to obtain the same information, the designer should choose which ones to monitor. The object that is the most appropriate to monitor will be different for different context elements and might be influenced by the available techniques and by practical limitations.

Illustration Step 2.2 (tour guide): In the tour guide example, we identified the location of the user, represented as $hasLocation(U, L_1)$, as a context element. It has two arguments that refer to objects, viz. the user and a location. A designer could choose to monitor the user and determine their location. However, an alternative is to monitor a location and establish what users are there. Nevertheless, to protect the privacy of the people at the locations that do not use the tour guide, the designer could choose to monitor the user.

Illustration Step 2.2 (B2G information sharing system): In the example of the B2G information sharing system, we identified the sensitivity of data from business A to business B, represented as $isSensitiveTo(Data, BusinessA, BusinessB)$, as a context element. Data is not inherently sensitive in this case, so it is not useful to monitor the data. Business B might know what data is too sensitive for them to have, but it is not in their interest to open up about this. However, business A will know exactly what data they do not what to share with what other businesses because of its sensitivity. Therefore, the context-aware system should monitor businesses to establish what data they consider sensitive from what other parties.

A designer who wants to improve the accuracy of the context information might choose to monitor more than one type of object for a context element; for instance, they might monitor both the user and the location. However, it is important to note that this will come at additional costs, as it requires the addition of some kind of conflict resolution for dealing with contradictory sensor information.

Once an object to monitor has been chosen, the designer needs to decide how to measure what it is connected to according to the relationship in the context element. It is important to be aware that it is often useful to use the same

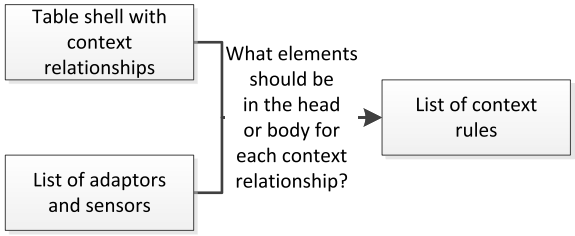

Fig. 9. Determining the rules for reasoning with context information for a context-aware system.

type of measurement to measure things that are similar. In the tour guide example, it would make sense to use the same measurements for the location of the user and for the location of the sight.

The next step is identifying what kind of sensor components could provide the measurement and how this could be done. The designer should provide a description of these sensor components and include them in the overall architecture. Like the adaptor components, the sensor components belonging to a sensor element should connect to all objects in that element.

Illustration Step 2.2 (tour guide): For the location context element it was decided to monitor the user. We now need to find a measurement for the location of the user. We could choose coordinates for this, as this is quite a common measurement for location. A very common way to determine coordinates is to use a GPS sensor. This would thus be an appropriate sensor to measure the location of the user. To connect to the user, the GPS sensor should be placed on something that the user carries with them; this could be the tour guide itself. To connect to the location, the sensor performs its measurement.

Illustration Step 2.2 (B2G information sharing system): For the sensitivity context element, it was decided to monitor the business that thinks that the data is sensitive. We thus need to find a measurement for what data they consider sensitive, and from what businesses. For the businesses, we can assign IDs to the data that is shared and we can name businesses. The most obvious way to sense what data is sensitive and to what businesses, is to ask the business that thinks the data is sensitive. A sensor should thus request this data from businesses. In the case of the B2G information sharing system, the component simply requests the information from businesses when new data is shared.

In section 4.4, we discussed that in some cases relationships can be included that are not context elements, but express constraints. These relationships do not change over time. This means that to 'sense' them, no external connections need to be made. Instead, this information needs to be stored somewhere in the system itself or calculated. For instance, for the distance between locations, the 'sensor' could be a component that calculates the distance between two coordinates.

5.3. Step 3: determining the rules for reasoning with context information

The overall process of performing step 3 is shown in Fig. 9. The inputs of this step are the outputs of steps 1 and 2, namely a list of context relationships and a list of adaptors and sensors. The output of this step is a list of context rules that the context-aware system can use to derive what adaptations to make in different situations.

The input of the reasoning component of a context-aware system comprises information gathered by the sensors. They are expressed as ground literals in a logic program. This might require the raw data of the sensor elements to be translated into these literals by middleware. There are ample descriptions in the literature of how middleware can be used to process raw context information (for an overview, see e.g. [73]).

The outputs of the reasoning component also are ground literals. They express the value that the adaptor elements should have to ensure the appropriate value of the focus. They are input for the adaptors and can be viewed as commands to perform an action that results in the adaptor element being true. For this, the middleware between the reasoning component and the adaptors themselves should include a mapping between literals and the actions of adaptors.

A context rule is a rule that expresses that the system needs to perform a manipulation to make the environment element in its header true in the situation that the environment elements in its body are true. The syntax of a context rule can be found in Definition 2. Examples of context-rules can be found in Example 2.

The context relationships provide information on what manipulations we want to perform in what situations. As discussed, for the positive context relationships, we want the situation described for them to exist, and for the negative context relationships, we want the situation described for them not to exist. Based on this principle, we already established what context elements should be manipulated in step 2.1.

We can translate each positive context relationship into a context rule where the head is a schematic literal representing the required value of one of its adaptor elements, and the body is the set of all schematic literals representing the values of its relationships that are not adaptor elements. The number of different rules there are for a positive context relationship is the number of adaptor elements it has. A designer should derive all possible rules for each positive context relationship in this way.

Illustration Step 3 (tour guide): The context relationship in Table 5 can be translated into the following context rule.

$$
\begin{aligned}
&isProvidedTo\,(I, U) \leftarrow \\
&isUser\,(U)\,, \\
&subjectOf\,(S, I)\,, \\
&hasLocation\,(U, L_1)\,, \\
&hasLocation\,(S, L_2)\,, \\
&distance\,(L_1, L_2, Distance)\,, \\
&Distance < 150
\end{aligned}
$$

We cannot translate this context relationship into any other rules, as it has only one adaptor element.

We can translate every negative context relationship into a single rule, where the body is again the set of all literals representing values for context elements in its situation that are not adaptor elements. As negative context relationships have only one adaptor element, and we do not want the situation in the negative context relationships to exist, the head of the rule is the negation of the schematic literal representing the value of the adaptor element in the situation described in the context relationship. The negation is that of classical logic, in the sense that the double negation of a literal is equivalent to the literal.

Illustration Step 3 (B2G information sharing system): The context relationship in Table 6 can be translated into the following context rule.

$$
\begin{aligned}
&\neg isSharedWith\,(Data, BusinessB) \leftarrow \\
&isBusiness\,(BusinessA)\,, \\
&isBusiness\,(BusinessB)\,, \\
&isSensitiveTo\,(Data, BusinessA, BusinessB)
\end{aligned}
$$

To complete step 3, the designer should generate all context rules in this way. These context rules can then be reasoned with in a logic program such as described by Lifschitz [66], together with the context elements that are input for the reasoning component. From the logic program it can be derived what manipulations the system should perform in a certain situation. Because we used the logic programming paradigm for our formalisation, the context rules and context elements are in the appropriate format to be part of the logic program.

We will illustrate the workings of such a logic program based on the simplified logic program in Example 4:

$$
\begin{aligned}
&\{isProvidedTo\,(I, U) \leftarrow \\
&isUser\,(U)\,, isInformation\,(I)\,, subjectOf\,(S, I)\,, isSight\,(S)\,,\ hasLocation(U, L_1\,), \\
&hasLocation(S, L_2\,), distance(L_1, L_2, Dist), Dist < 150,
\end{aligned}
$$

$$
\begin{aligned}
&isUser\,(mary)\,, \\
&isSight\,(pyramidOfCheops)\,, \\
&isInformation\,(info)\,, \\
&hasLocation\,\big(mary, 29°58'45.03''\text{N}\ 31°08'03.69''\text{E}\big)\,, \\
&hasLocation\,\big(pyramidOfCheops, 29°58'27.00\text{N}\ 31°08'2.21\text{E}\big)\,, \\
&subjectOf\,(pyramidOfCheops, info)\}.
\end{aligned}
$$

The ground literals in the logic program express the context information that the program receives from its sensors as input. Context information expressed in the example is that *mary* is a user and that her location is $29°58'45.03''$N $31°08'03.69''$E. Only if the situation is what it should be according to the body of a context rule, the head of the rule can be derived and a command to perform an action is provided. In this way, the logic program is used to derive the actions that need to be performed by the adaptors of the system based on context information.

In this example, it can be derived that the location of *mary* is less than 150 metres from the subject of *info*, namely *pyramidOfCheops*, by substituting U for *mary*, and L_1 for $29°58'45.03''$N $31°08'03.69''$E, and so on. This means that the head of the rule can be derived and a command is provided to an adaptor to provide *info* to *mary*.

Of course, the ground literals in the program can be derived as well. However, they are already true, so no action needs to be performed to make them true. It would be easy to let them be filtered out by the middleware of the system by comparing the input and the output of the reasoning component.

Furthermore, the rule provided here includes some restrictions that are not context elements and that cannot be sensed or manipulated. This is for example the case for $distance(L_1, L_2, Dist)$. Evaluation of whether these literals are true can be easily done by adding some standard ground literals that never change (e.g. for $hasLocation(pyramidOfCheops$, $29°58'27.00\text{N}31°08'2.21\text{E})$), or by calculating them using functionality that are usually build-in in a programming language (e.g. for $Dist < 150$).

In general, logic programs will be quite simple, because only a single rule, rather than a sequence of rules, is needed to derive a manipulation (not taking into account calculating the restrictions). However, what specific variety of logical program and corresponding reasoning mechanism will be chosen is up to the designer as there are a variety of practical

factors that might play a role. For instance, a context-aware system that should respond to changes in the environment very fast and that contains only a couple of rules, could benefit from a reasoning mechanism that derives all manipulations in a bottom-up approach each time new sensor information is available. On the other hand, the designer of a system that contains a lot of rules and facts might prefer a top-down approach in which the adaptors periodically query the system for the next manipulation that they need to perform. Furthermore, whether negation as failure is enough in the body of rules could depend on the required evidence based on which a manipulation needs to be derived.

It is important to note that it is possible that multiple values for environment elements for the same object are derived from the rules. Adaptors will not always be equipped to cope with these multiple values simultaneously (e.g. screen off and on at the same time). Furthermore, if the heads of the rules contain classical negation, it is possible to derive contradictions that no adaptor could conform to. For instance, when based on one rule $isProvidedTo(info, mary)$ is derived and when based on another rule $\neg isProvidedTo(info, mary)$ is derived. It is not possible to provide $info$ to Mary and not provide $info$ to Mary at the same time.

The reason for these issues is that for the more complex cases, the insight into context from step 1 is typically incomplete. In section 5.1.2 we described that in practice it is not feasible to establish what comprises the complete set of environment elements that have a certain context relationship with the focus. This is not an issue specific to the method or definitions we propose; rather, it is a fundamental issue when investigating complex environments in the real world. There are just too many variables to take into account. In fact, defeasible logics were developed to deal with this same issue. Defeasible logic programs, for example as described by García and Simari [76], could also be used to reason with the rules.

Alternatively, this issue can be dealt with in the middleware between the adaptors and the reasoning mechanism. There are many ways to solve this, and again, the best way depends on a lot of practical considerations. In some cases it might, for instance, be useful to ask the user to choose between alternative manipulations. In other cases, it might be better not to bother the user with this and to implement an algorithm that makes a choice between alternative manipulations. If necessary, such an algorithm could also get insight into the rules and the facts that manipulations were derived from.

6. Conclusions and suggestions for further research

The development of the method was driven by a practical need we experienced in a project in which we developed a context-aware B2G information sharing system in the highly complex domain of international container shipping. While the literature on context-aware systems is extensive, there is no systematic approach to get insight into context and to use this insight to design context-aware systems in complex environments with, for example, many actors and in which legislations might have an influence. Such an approach is necessary to ensure the efficiency and effectivity of the design process. Current work too often has to rely on assumptions about what belongs to context and how the system should adapt to context. This is associated with risks of ambiguity and not taking into account the appropriate context in the design of the system. To address this problem, in this paper we proposed a method that provides a way to structure the investigation of context and to clearly distinguish irrelevant elements in the environment from relevant context elements. It also provides steps to derive the design of a context-aware system from the insight into context.

The method describes how designers can investigate context and collect the information they need to design a context-aware system. Furthermore, designers can use the method to directly establish what sensors and adaptors should be included. In addition, we provide a way for designers to translate the knowledge on context into rules that the system can use to determine what adaptations need to be made. The method thus consists of three steps: 1) getting insight into context, 2) determining the components needed to sense context and adapt to it, and 3) determining the rules according to which the system should adapt. Step 1 is critical, as steps 2 and 3 directly depend on it. When a designer has followed the steps, they can further work out the technical details of the system using existing tools and frameworks that are covered in the literature.

Concerning the effectiveness of the design process, we checked the context elements and relationships we found with researchers with domain expertise. Based on this, we only had to make minor additions and changes to the context elements and relationships found.

The method is based on a criterion for determining whether something is a context element and a way to quickly check whether the criterion is met. The structured manner in which data on context is gathered and analysed when the method is used, also helps to make it explicit when this is ambiguous. For instance, it is easy to establish that there is a problem if the same situation has different impacts on the focus according to different data sources. When such a conflict is found, a designer can concentrate their efforts on solving it. They could do so by, for example, finding out whether the conflict results from not including enough context elements in the situation description, or by additional checking of whether their data sources are reliable.

In complex environments, a variety of parties are involved in information sharing. Efficiency of the design process will also be influenced by the interactions with these parties. We explained the notions of ‘context element’ and ‘context relationship’ to domain experts and juridical experts we interviewed to obtain insight into context. The interviewees all indicated that they understood what was meant by these notions. Only in a few cases in the beginning of the interviews they started to use these notions spontaneously themselves. However, each of the interviewees seemed to concentrate on finding things that impact the focus. This might be due to the use of these concepts in the introduction of the interviews. In addition, one of the lawyers we collaborated with indicated that using these concepts helped her to understand what

we were looking for and to structure the information obtained on context. Furthermore a researcher with expertise in the B2G information sharing domain stated that she believed that the structuring that the notions of 'context element' and 'context relationship' offer, help to provide scope to discussions. In addition, she states that it provides a shared vocabulary facilitating discussions with the various organisations.

An underlying assumption for the method is that the appropriate elements to take into account in the design of a context-aware system are those that should be sensed or manipulated in order to reach the design goal. Interpreting 'appropriateness' in this way, the use of the method reduces the risk of not taking into account the appropriate context in the design of the context-aware system. That is, the method supports systematic data collection on context elements and context relationships. Furthermore, once data is gathered, the appropriate sensors and adaptors can be identified quite straightforwardly based on the data. In addition, the data gathered on the context relationships can be translated into rules for the system. Thus, the method supports not only obtaining the necessary insight to take the appropriate context into account, but also basing a design on this insight.

One issue that might make it difficult to take the appropriate parts of the context into account is that the search space for things that belong to context can be very large. As we are dealing with complex and open environments, it is impossible to guarantee that all context elements will be found by a designer using the method. However, the method might make the search process more efficient by supporting the designer in specifying what exactly they are looking for, by letting them determine and specify their focus. The method then provides guidance on how to use the focus to select appropriate data sources and how to use the focus to determine which parts of the gathered data to focus on.

The focus, in a sense, thus determines the scope of the context-aware system and what is taken into account as context. This means that it is of paramount importance for a designer to consider and choose their focus carefully. The usual procedure for designing a system is to start with a specification of a goal, or a problem to solve. The focus is directly derived from this and determining the focus therefore is unlikely to require a lot of additional effort by the designer. What the new method adds to this procedure is that it makes it possible to link parts of the environment with the focus, thereby providing a designer with an easier way to determine what belongs to their scope, or in other words, the context that they should take into account for their design.

The use of a focus to select context elements and thereby sensors and adaptors means that several things that are usually not considered context, context-aware systems or sensors can be included. For example, the 'traditional' notion of sensor would refer to devices measuring things like GPS location or temperature. Using the method, it could also refer to organisations, for example. We believe that this deviation from the traditional interpretation of context and associated notions is due to the practice being leading instead of the literature. Furthermore, we believe that from a pragmatic point of view, it is useful to extent these notions in this way. Including businesses as sensors, for example, is clearly necessary in the case of the B2G information sharing system as there is no traditional sensor that can measure things like the relationships between businesses. Information on this is needed for the system to reason with and adapt to, which are things that are very typical for context-aware systems.

In addition, future research could focus on the conflict resolution mechanism for dealing with incompatible manipulations. First of all, it needs to be determined whether conflict resolution should happen when generating the rules, when reasoning with the rules or in the middleware between the reasoning component and the adaptors. Second, it seems that this conflict resolution should rely, at least partially, on whether a rule was derived from a positive context relationship or a negative one. When it is not possible to perform one of the manipulations that stem from a positive context relationship, the desired situation cannot be produced. It might then not be useful to try and fulfil the others. Future research should determine how to deal with this issue. In addition, further research should focus on determining whether the use of literals and context rules for the system, as generated in our method, are expressive enough to also be used in other domains.

Acknowledgements

The work in this paper is part of the project JUridical and context-aware Sharing of informaTion for ensuring compliance (JUST). This project is funded by the Netherlands Organisation for Scientific Research (NWO) as part of the ISCOM programme (Innovation in Supply Chain Compliance and Border Management) under grant number 438-13-601. We would like to thank dr. Helle Hansen for her advice on formalising the definition of context and related concepts.

References

[1] W. Schwinger, C. Grun, B. Proll, W. Retschitzegger, A Schauerhuber, Context-awareness in mobile tourism guides – a comprehensive survey, in: Handb. Res. Mob. Multimedia, vol. 2, second ed, 2007, pp. 298–314.
[2] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (IoT): a vision, architectural elements, and future directions, Future Gener. Comput. Syst. 29 (7) (2013) 1645–1660.
[3] X. Fang, S. Misra, G. Xue, D. Yang, Smart grid – the new and improved power grid: a survey, IEEE Commun. Surv. Tutor. 14 (4) (2012) 944–980.
[4] A. Whitmore, A. Agarwal, L. Da Xu, The Internet of things a survey of topics and trends, Inf. Syst. Front. 17 (2) (2015) 261–274.
[5] L. Atzori, A. Iera, G. Morabito, The Internet of things: a survey, Comput. Netw. 54 (15) (2010) 2787–2805.
[6] J.-Y. Hong, E.-H. Suh, S.-J. Kim, Context-aware systems: a literature review and classification, Expert Syst. Appl. 36 (4) (2009) 8509–8522.
[7] G.D. Abowd, A. Dey, P. Brown, N. Davies, M. Smith, P. Steggles, Towards a better understanding of context and context-awareness, in: Handheld and Ubiquitous Computing, 1999, pp. 304–307.
[8] B.N. Schilit, M.M. Theimer, Disseminating active MapInformation to mobile hosts, IEEE Netw. 8 (5) (1994) 22–32.

[9] A.K. Dey, G.D. Abowd, Towards a better understanding of context and context-awareness, Comput. Syst. 40 (3) (1999) 304–307.
[10] D. Hesketh, Weaknesses in the supply chain: who packed the box, World Cust. J. 4 (2) (2010) 3–20.
[11] B. Klievink, M. Janssen, Y.-H. Tan, A stakeholder analysis of business-to-government information sharing, Int. J. Electron. Gov. Res. 8 (4) (2012) 54–64.
[12] B. Klievink, G. Zomer, IT-enabled resilient, seamless and secure global supply chains: introduction, overview and research topics, Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 9373 (2015) 443–453.
[13] T. Jensen, Y.-H. Tan, Key design properties for shipping information pipeline, in: Open and Big Data Management and Innovation, 2015, pp. 491–502.
[14] Y.-H. Tan, N. Bjørn-Andersen, S. Klein, B. Rukanova, Accelerating Global Supply Chains with IT-Innovation, 2011.
[15] M.C. Cooper, L.M. Ellram, J.T. Gardner, A.M. Hanks, Meshing multiple alliances, J. Bus. Logist. 18 (1) (1997) 67.
[16] H. Min, G. Zhou, Supply chain modeling: past, present and future, Comput. Ind. Eng. 43 (1–2) (2002) 231–249.
[17] D.M. Lambert, M.C. Cooper, J.D. Pagh, Supply chain management: implementation issues and research opportunities, Int. J. Logist. Manag. 9 (2) (1998) 1–19.
[18] Z. Kenessey, The primary, secondary, tertiary and quaternary sectors of the economy, Rev. Income Wealth 33 (4) (1987) 359–385.
[19] P.A. Dabholkar, S.M. Neeley, Managing interdependency: a taxonomy for business to business relationships, J. Bus. Ind. Mark. 13 (6) (1998) 439–460.
[20] T. Jensen, N. Bjørn-andersen, R. Vatrapu, Avocados crossing borders: the missing common information infrastructure for international trade, Cult. Int. Context (2014) 15–24.
[21] T. Jensen, R. Vatrapu, Ships & roses: a revelatory case study of affordances in international trade, in: Proceedings of ECIS 2015, 2015, pp. 1–18.
[22] C. Bolchini, F.A. Schreiber, L. Tanca, A methodology for a very small data base design, Inf. Syst. 32 (1) (2007) 61–82.
[23] R. Casas, D. Cuartielles, Á. Marco, H.J. Gracia, J.L. Falcó, Hidden issues in deploying an indoor location system, IEEE Pervasive Comput. 6 (2) (2007) 62–69.
[24] J. Anhalt, A. Smailagic, D.P. Siewiorek, F. Gemperle, D. Salber, S. Weber, J. Beck, J. Jennings, I.B.M.T.J. Watson, Toward context-aware computing: experiences and lessons, IEEE Intell. Syst. 16 (3) (2001) 38–46.
[25] I. Augustin, A.C. Yamin, L.C. Da Silva, R.A. Real, G. Frainer, C.F.R. Geyer, ISAMadapt: abstractions and tools for designing general-purpose pervasive applications, Softw. Pract. Exp. 36 (11–12) (2006) 1231–1256.
[26] D. Salber, A.K. Dey, G.D. Abowd, The context toolkit: aiding the development of context-enabled applications, in: Proceedings of the Conference on Human Factors in Computing Systems, 1999, pp. 434–441.
[27] J. Hong, J. Landay, An infrastructure approach to context-aware computing, Hum.-Comput. Interact. 16 (2) (2001) 287–303.
[28] Q. Wei, K. Farkas, C. Prehofer, P. Mendes, B. Plattner, Context-aware handover using active network technology, J. Comput. Netw. 50 (15) (2006) 2855–2872.
[29] L. Qiu, L. Chang, F. Lin, Z. Shi, Context optimization of AI planning for semantic web services composition, Serv. Oriented Comput. Appl. 1 (2) (2007) 117–128.
[30] M.J. van Sinderen, A.T. van Halteren, M. Wegdam, H.B. Meeuwissen, E.H. Eertink, Supporting context-aware mobile applications: an infrastructure approach, IEEE Commun. Mag. 44 (9) (September 2006) 96–104.
[31] K. Henricksen, J. Indulska, Developing context-aware pervasive computing applications: models and approach, Pervasive Mob. Comput. 2 (1) (2006) 37–64.
[32] U. Alegre, J.C. Augusto, T. Clark, Engineering context-aware systems and applications: a survey, J. Syst. Softw. 117 (2016) 55–83.
[33] A. Finkelstein, A. Savigni, A framework for requirements engineering for context-aware services, in: First International Workshop from Software Requirements to Architectures, STRAW'01, 2001, pp. 36–41.
[34] B. Nuseibeh, S. Easterbrook, Requirements engineering: a roadmap, in: Proceedings of the Conference on the Future of Software Engineering, ICSE '00, 2000, pp. 35–46.
[35] D.M. Berry, B.H.C. Cheng, J. Zhang, The four levels of requirements engineering for and in dynamic adaptive systems, in: 11th International Workshop on Requirements Engineering: Foundation for Software Quality, REFSQ'05, 2005, pp. 113–120.
[36] P. Sawyer, N. Bencomo, J. Whittle, E. Letie, A. Finkelstein, Requirements-aware systems: a research agenda for RE for self-adaptive systems, in: Proceedings of the 2010 18th IEEE International Requirements Engineering Conference, RE2010, 2010, pp. 95–103.
[37] W. Sitou, B. Spanfelner, Towards requirements engineering for context adaptive systems, in: Proceedings 0f the International Computer Software and Applications Conference, vol. 2, 2007, pp. 593–598.
[38] L. Kolos-Mazuryk, G.-J. Poulisse, P. van Eck, Requirements engineering for pervasive services, in: OOP-SLA'05 Workshop on Creating Software for Pervasive Services, 2005, pp. 1–5.
[39] A.R. Hevner, A three cycle view of design science research, Scand. J. Inf. Syst. 19 (2) (2007) 87–92.
[40] A.R. Hevner, S.T. March, J. Park, S. Ram, Design science in information systems research, MIS Q. 28 (1) (2004) 75–105.
[41] K. Peffers, T. Tuunanen, M.A. Rothenberger, S. Chatterjee, A design science research methodology for information systems research, J. Manag. Inf. Syst. 24 (3) (2007) 45–77.
[42] M. Perttunen, J. Riekki, O. Lassila, Context representation and reasoning in pervasive computing, Int. J. Multimed. Ubiquitous Eng. 4 (4) (2009) 1–28.
[43] T. Strang, C. Linnhoff-Popien, A context modeling survey, in: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 – The Sixth International Conference on Ubiquitous Computing, 2004, pp. 1–8.
[44] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, D. Riboni, A survey of context modelling and reasoning techniques, Pervasive Mob. Comput. 6 (2) (2010) 161–180.
[45] K.S. Sagaya Priya, Y. Kalpana, A review on context modelling techniques in context awarecomputing, Int. J. Eng. Technol. 8 (1) (2016) 429–433.
[46] T. Winograd, Architectures for context, Hum.-Comput. Interact. 16 (2) (2001) 401–419.
[47] P. Benou, C. Vassilakis, V. Costas, C. Vassilakis, V. Costas, C. Vassilakis, The conceptual model of context for mobile commerce applications, Electron. Commer. Res. 10 (2) (2010) 139–165.
[48] J.W. Kaltz, J. Ziegler, S. Lohmann, Context-aware web engineering: modeling and applications, Rev. Intell. Artif. 19 (3) (2005) 439–458.
[49] J.L. Crowley, Context driven observation of human activity, in: European Conference on Ambient Intelligence, in: in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 2875, 2003, pp. 101–118.
[50] A. Zimmermann, A. Lorenz, R. Oppermann, S. Augustin, An operational definition of context, in: CONTEXT'07: Proceedings of the 6th International and Interdisciplinary Conference on Modeling and Using Context, in: LNAI, vol. 4635, 2007, pp. 558–571.
[51] Y.-K. Wang, Context awareness and adaptation in mobile learning, in: 2nd IEEE International Workshop on Wireless and Mobile Technologies in Education, WMTE'04, 2004, pp. 154–158.
[52] J.L. Crowley, J. Coutaz, P. Reigner, Perceptual components for context aware computing, in: International Conference on Ubiquitous Computing, 2002, pp. 117–134.
[53] Z. Yang, Z. Qilun, L. Fagui, An extended context model in a rfid-based context-aware service system, in: Proceedings of the 2nd 2008 International Symposium on Intelligent Information Technology Application Workshop, IITA 2008 Workshop, 2008, pp. 693–697.
[54] K.K. Khedo, Context-aware systems for mobile and ubiquitous networks, in: Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, ICNICONSMCL'06, 2006, pp. 123–130.
[55] P. Dourish, What we talk about when we talk about context, Pers. Ubiquitous Comput. 8 (1) (2004) 19–30.

[56] P. Brézillon, Task-realization models in contextual graphs, in: 5th International and Interdisciplinary Conference, CONTEXT 2005, 2005, pp. 55–68.
[57] V. Vieira, P. Tedesco, A.C. Salgado, Designing context-sensitive systems: an integrated approach, Expert Syst. Appl. 38 (2) (2011) 1119–1138.
[58] P. Brézillon, J. Pomerol, Contextual knowledge sharing and cooperation in intelligent assistant systems, Trav. Hum. 62 (3) (1999) 223–246.
[59] A.R. Hevner, S. Chatterjee, Design Research in Information Systems, Integrated Series in Information Systems, vol. 28, 2004, pp. 75–105.
[60] S.T. March, G.F. Smith, Design and natural science research on information technology, Decis. Support Syst. 15 (4) (1995) 251–266.
[61] J. Pries-Heje, R. Baskerville, J. Venable, Strategies for design science research evaluation, in: ECIS 2008 Proc., 2008, pp. 1–12.
[62] R. Kowalski, Predicate logic as a programming language, in: IFIP Congress, 1974, pp. 569–574.
[63] D.H.D. Warren, L.M. Pereira, F. Pereira, Prolog – the language and its implementation compared with Lisp, ACM SIGART Bull. 64 (12) (1977) 109–115.
[64] A. Colmerauer, Les grammaires de métamorphose, 1975.
[65] M.H. Van Emden, Programming with Resolution Logic, 1975.
[66] V. Lifschitz, Foundations of logic programming, in: G. Brewka (Ed.), Principles of Knowledge Representation 3, CSLI Pub., 1993, pp. 69–127.
[67] S. van Engelenburg, M. Janssen, B. Klievink, What belongs to context? A definition, a criterion and a method for deciding on what context-aware systems should sense and adapt to, in: Software Engineering and Formal Methods 2017, in: LNCS, vol. 10729, 2017, pp. 101–116.
[68] "Context," Oxford Dictionaries. [Online]. Available, https://en.oxforddictionaries.com/definition/context. (Accessed 27 July 2018).
[69] C. Goodwin, A. Duranti, Rethinking context: an introduction, in: Rethinking Context: Language as an Interactive Phenomenon, Cambridge University Press, 1992, pp. 1–42.
[70] A.K. Dey, Understanding and using context, Pers. Ubiquitous Comput. 1 (5) (2001) 4–7.
[71] H.A. Simon, The Sciences of the Artificial, vol. 33, no. 5, MIT Press, 1996.
[72] B. Rettler, A.M. Bailey, Object, in: The Stanford Encyclopedia of Philosophy, 2017, pp. 1–22.
[73] M. Baldauf, S. Dustdar, F. Rosenberg, A survey on context-aware systems, Int. J. Ad Hoc Ubiq. Comput. 2 (4) (2007) 263.
[74] R. Schmohl, U. Baumgarten, M. D. -G, A generalized context-aware architecture in heterogeneous mobile computing environments a generic context-aware architecture, in: The Fourth International Conference on Wireless and Mobile Communications, 2008, pp. 118–124.
[75] A. Saeed, T. Waheed, An extensive survey of context-aware middleware architectures, Am. J. Comput. Archit. 1 (3) (2012) 51–56.
[76] A.J. García, G.R. Simari, Defeasible logic programming an argumentative approach, Theory Pract. Log. Program. 4 (2) (2004) 95–138.