Supporting the Standardization Process of Smart City Systems

Marion Gottschalk
OFFIS - Institute for Information Technology, Escherweg 2, 26121 Oldenburg, Germany

1 RESEARCH PROBLEM

The continuous development in information and communication technologies (ICT) and the resulting research area Internet of Things (IoT) enable the connection of more and more hardware and software systems (DaCosta, 2013). IoT describes a global infrastructure for the information society to enable interconnecting physical and virtual things based on existing and evolving ICT (ITU-T, 2012). Regarding the development of smart cities that constitutes a system of systems concept using ICT (DKE, 2015), a defined, or, better still, a standardized way to define and to link various systems is necessary.

Smart cities are innovative cities that use ICT to make urban operations and services more efficient for improving the quality of life (ITU-T FG-SSC, 2014). An example or rather a use case of a smart city system can be "Supporting the traffic flow in cities" (Tragos et al., 2014). The use case describes relevant requirements for all involved components and systems as well as the communication between different ICT that can be used for a real time traffic estimation in cities. For example, traffic cameras can be used for recording the current traffic volume and analyzing the data with an analysis software. This data has to be sent within two seconds from the traffic cameras to the analysis software for switching lights in time to keep the traffic flowing.

Nowadays, various companies are developping new components or systems that can be part of a smart city. However, components from different companies are often incompatible with each other. For example, Siemens developed a traffic control system on the basis of their private cloud. The system uses control units that are installed on single lights to control these with smart phones or tablets via a secure connection (Wagner, 2014). Another example is the traffic incident management solution by Cisco and AGT International (CISCO and AGT International, 2014). They use various sensors (e.g. traffic cameras and in-pavement vehicle sensor) to analyze the current traffic volume and to enable a smooth traffic flow as well as to detect incidents. These companies are likely to define their own interfaces for single components, e.g. for their software systems like the Siemens private cloud. These interfaces enable the interconnection of their components and software. Reasons for diversity are e.g. market power, protection of ideas, or delimitation to other products. Standardization organizations try to unify the development of smart city components and systems by defining and standardizing interfaces. For example, it is helpful to create generic use cases for describing functional requirements that show the general interaction between two components, such as city buses and traffic lights (Tragos et al., 2014), and their interfaces needed. This kind of use cases can be created by summarizing individual use cases, like the functional description by the Siemens traffic control system and the Cisco and AGT International traffic incident management system, to get one generic use case for supporting the traffic flow. If the generic use cases are public, this approach will support other companies to adapt their components and systems for existing implementations.

Smart cities are difficult to realize because they are complex based on their system of systems concept. They are not implemented in one step, because cities have to be evolve slowly to smart cities (DKE, 2015). It already starts with the specification process. Each smart city component or system has many functional and non-functional requirements that have to be matched with each other and have to be complemented with further requirements to connected components so-called interface requirements. One aim for standardization organizations is the consistent description of smart city requirements to get components and systems by different vendors that are interoperability (DKE, 2015). On the basis of these requirements, smart city components and systems can be implemented. If errors occurred or are detected during the implementation process, it can be very time consuming and expensive to correct them (Sommerville, 2009). On the one hand, it costs much time to detect the reason for errors. On the other hand, errors frequently influence other functions maybe other components which cause further errors. In addition, experiences on prior projects have

Gottschalk, M.
Supporting the Standardization Process of Smart City Systems.
In *Doctoral Consortium (DCSMARTGREENS 2016)*, pages 17-27

shown that errors during the implementation process often base on faulty requirements and using formal methods in the requirements specification process can reduce them (Sommerville, 2009). Hence, it is important and sensible for standardization organizations to invest time and money in the specification process of smart cities to get consistent requirements and to prevent errors. The following research question was identified: *"How can the management and collection of complex hardware and software requirements be supported to get syntactically correct and consistent requirement specifications?"*

Therefore, the research problem is to check the described requirements for their syntactical correctness and consistency automatically or at least semi-automatically. Therefore, various techniques from the requirements specification process can be used; however, which techniques are suitable for the standardization process, and for the complex requirement specification of smart cities in general. In Section 2, the resulting objectives for this work are described in more detail. Section 3 gives an overview of requirements techniques and further basics, which are important for this work. Building on this, the procedure and methodology for the PhD project follows in Section 4. The evaluation process and the expected results are demonstrated in Section 5. Finally, the stage of the PhD project is shown in Section 6.

2 OUTLINE OF OBJECTIVES

Aim of the PhD project is to review different techniques for checking requirements syntactical correctness and consistency as well as to apply these on complex requirement specifications. According to the research question and the main goal, following objectives for the PhD project can be defined:

- visualizing requirements: the graphical representation of requirements and their relations supports requirement engineers to get a better overview of existing requirements. hence, visualization techniques are also considered to create consistent requirement specifications.
- reducing syntactically incorrect requirements: an analysis on requirement specifications only works if the syntax of textual and formal requirements is correct; therefore, the analysis starts with an syntax check.
- reducing inconsistent requirements: the development of a system with inconsistent requirements often fails and leads to high costs for correcting them; thus, projects should be supported from the beginning of the specification process.
- implementing an approach for consistency checks: considered techniques for creating syntactically correct and consistent requirements are used to implement a new approach or combine known approaches for checking requirement specifications.
- evaluating the approach: the implemented approach should be checked in the context of smart city, smart grid, and electrical mobility; therefore, described requirements in various project and standardization groups are used.

If these objectives are reached, the work will have benefits and relevance for various people and groups. A syntactically correct and consistent requirements specification supports the full development process for each system and improves its quality from the beginning. It supports requirement engineers, system architects, and customers or experts who set requirements for creating a good basis for system designs. Additionally, costs for developing systems are lower when errors are avoided during the requirements specification process.

3 STATE OF THE ART

The research problem comprises two main research fields: smart cities and requirement specifications for a consistent description of future cities. In addition, the use case methodology from the IEC and related work are presented to demonstrate the current development in these fields.

3.1 Smart Cities

As mentioned above, smart cities are systems of systems (SoS); they include sub-systems like facility management, smart home, as well as mobility and logistics (DKE, 2015). Figure 1 shows an overview of a smart city concept. It demonstrates the mentioned sub-systems, which are based on the development of the smart city infrastructure architecture model (SCIAM) (DKE, 2014). An architecture model is a graphical representation of a use case that shows business and technical aspects at a glance. The connection of smart city sub-systems have to be done in a step-by-step implementation to exchange pre-existing systems by new ones. Urban operations and services are more efficient and the quality of life is improved after interconnecting these systems (ITU-T FG-SSC, 2014). The International Telecommunications Union (ITU) defines a smart city as *"an innovative city that uses ICTs and other means to improve quality of life,*

efficiency of urban operation and services, and competitiveness, while ensuring that it meets the needs of present and future generations with respect to economic, social, environmental as well as cultural aspects" (ITU-T FG-SSC, 2014). This definition includes technical, connecting sub-systems via ICT, as well as external aspect, the comprehension of economic, social, environment, and culture topics, that should increase the acceptance of all participants in smart cities. Thus, the planning and implementation of smart cities shall comply with this definition.

Figure 1: Smart City Concept.

However, the complexity, step-by-step implementation, and interconnection with external aspects results in difficulties, such as connecting with existing systems and matching data protection laws, that have to be solved before systems like smart cities can be implemented. Such difficulties are already described by Rittel and Weber (Rittel and Webber, 1973) as dilemmas in a general theory of planning in the year 1973. They describe the importance for planing systems and their desired behavior in detail. In addition, they consider changing systems and laws that make it difficult to describe and design a future SoS like smart cities completely, i.e. functionalities and difficulties have to be described and re-solved repeatedly during the implementation process (Rittel and Webber, 1973). Thus, this dilemma should be taken into account for the planning and development process of smart cities, and requirements as well as standards should be described in a way which allows adjustments and simple checks. Considering the smart city sub-systems and the involvement of humans, smart cities can be also seen as security-critical systems (DKE, 2014; Johnson, 2011), as well as cyber-physical systems from the view of the interconnected sub-systems (Wiesner et al., 2015).

3.2 Standards in the Context of Smart Cities

Various organizations, not only standardization organizations, address the planning and development of standards (DKE, 2015). Standardization organizations on different levels (international and national level) organize various working groups to discuss single topics in the smart city context. Some of the international standardization organizations are the International Organization for Standards (ISO), the International Electrotechnical Commission (IEC), and the International Telecommunication Union (ITU). The three organizations consider the development of smart cities from different views, the ISO addresses technical and economic aspects, the IEC deals with electrical, electric and related technologies, and the ITU considers telecommunication aspects. All these parts are important for the development of smart cities, and thus, it is necessary to arrange their experience that is already done in the *Smart City Study Group (SG 1)* (DKE, 2015). Further standardization organizations on European (e.g. European Committee for Standardization (CEN), European Telecommunication Standards Institute (ETSI)) or US (e.g. National Institute of Standards and Technology (NIST)) level deal with the topic *smart cities* and provide their results for the international organizations. The same applies for national organizations (e.g. The German Institute for Standardization (DIN) and The German standardization organization for Electrical, Electronic, and Information Technology (DKE)), their findings are also discussed on the next higher level. Another organization, involved in the standardization process, is the Institute of Electrical and Electronic Engineers (IEEE), which defines specifications regarding various topics that are referenced or used by standardization organizations. These organizations are examples, further exist. Due to the high number of various organizations, collecting consistent requirements and standards is difficult and have to be supported.

The importance for describing well-defined requirements to standardize smart city functionalities can be compared with the requirements specification process for SoS. Moreover, Sommerville et al. (Sommerville et al., 2011) define SoS according to the US Department of Defense (DoD) definition (Systems and Software Engineering Directorate, 2008) as *"a set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities"*. The smart city definition by the ITU is similar to this one, they differ in their accuracy, because a smart city is only a special kind of SoS. Based on the definition of smart cities as SoS, techniques from the requirements specification process in software systems should be adaptable and are considered here.

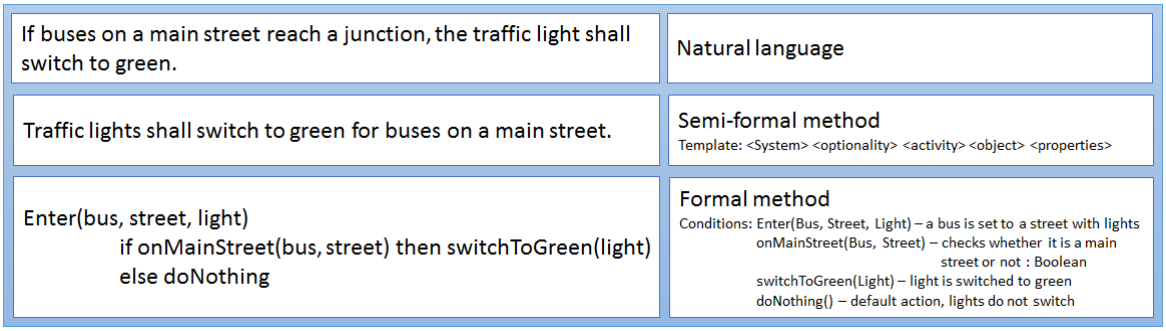

Figure 2: Smart City - Requirement Example.

3.3 Requirements Specification

The high relevance for well-defined requirements in SoS requires the viewing on various requirements' definitions and techniques. First, a general definition for requirements and possibilities is presented. Second, quality criteria for requirements are pointed out for demonstrating relevant aspects in the requirements specification process. One of these quality criteria is *consistency*. How to reach this is described in detail.

3.3.1 Definition and Examples

The term *requirements* is defined by the IEEE (IEEE, 1990) as follows: *"requirements are conditions or capabilities needed by a user to solve a problem or achieve an objective. These conditions and capabilities have to be met or possessed by a component or system to satisfy a contract, standard, specification or other formally imposed documents."* Thus, requirements describe functionalities and rules that are implemented by a component or system to meet user needs. A requirements list of a system should constitute its complete functionalities, but, it does not contain how these functionalities are implemented.

Requirements have to be written down to communicate different viewpoints of users, to check their consistency, and later, to proof whether the requirements are fulfilled. Therefore, various techniques exist that can be distinguished in three types: natural language, semi-formal methods, and formal methods (IEEE, 1990; Rupp, C. and die SOPHISTen, 2014). For each of these types, an example, based on the mentioned use case in Section 1 *Supporting the traffic flow in cities*, is shown in Figure 2. The requirement demonstrated is always the same. The natural language is the most frequently used method to describe requirements (Rupp, C. and die SOPHISTen, 2014), its rules are only based on the usage rather than being pre-established to the languages use (IEEE, 1990). The first requirement in Figure 2 shows the usage of natural language. Semi-formal methods are used for structuring natural language to facilitate its understanding; therefore, text pattern or templates can be used (Rupp, C. and die SOPHISTen, 2014). A simple template and an example for the semi-formal method is also shown in Figure 2. Formal methods often rely on mathematical, algebraic, or model-based representations to explain system functionalities (IEEE, 1990; Sommerville, 2009); therefore, variables for components, systems, and activities defined that are set in relation to each other. The last example in Figure 2 shows an algebraic representation of the requirement.

Which kind of requirements specification is used has to be decided for every project. The decision bases on the kind of system for which the requirements are defined. Security-critical systems are often described with formal methods to be sure whether all states of the system were considered and automatized analyses were executed. However, formal methods are hard to learn and time consuming, and thus not appropriately used (Sommerville, 2009; IEEE, 1998). Using natural language or semi-formal methods is easier for describing requirements, but, they are often ambiguous, which leads to mistakes in the development process (Rupp, C. and die SOPHISTen, 2014; IEEE, 1998). For describing requirements of smart cities as security- and safety-critical systems in detail, formal methods are useful for identifying interfaces and system states. However, general requirements on smart cities are often defined by experts without an ICT background, and thus, formal methods are undisclosed. Though, semi-formal methods like the use case methodology (IEC, 2015) are used for structuring requirements to facilitates the collection, sorting, and understanding of these, and it also allows simple analyses. Thus, requirements in natural and semi-formal language are considered in this work.

3.3.2 Quality Criteria

In the IEEE Recommended Practice for Software Requirements Specification (IEEE, 1998), quality criteria for requirements are defined as follows:

- correct: a requirement is correct if it matches the functionality that the systems shall implement. This cannot be checked by an automatized process, a review has to decide whether a requirement is correct or not.
- unambiguous: a requirement is unambiguous if it has only one interpretation, or rather, each part of the described system has its own uniques term that is used consistency during the project. Therefore, a glossary can be used that has to be used and checked by editors. Otherwise, the application of requirements specification languages detect many lexical, syntactic, and semantic errors that base on ambiguous statements.
- complete: requirements are complete if all functional as well as non-functional requirements of the planned system are collected.
- consistent: a requirement is consistent if no characteristic, logical, or temporal conflicts exist in itself. Additional, requirements have to be conflict-free between each other, this also means that requirements have to use the same terms to describe functionalities. Using the same terms makes it easier to detect same or conflicting requirements.
- ranked for importance and/or stability: requirements should be ranked, thus, each requirement has an identifier to indicate either the importance or stability. Some requirements may be essential, conditional, or optional for the final system, i.e. some requirements are necessary for a successful implementation and others are nice to have.
- verifiable: a requirement is verifiable if its successful implementation is measurable by a person or machine. Requirements that include terms like "works well" or "shall usually happen" are non-verifiable.
- modifiable: requirements are modifiable if any changes on them can be done easily, completely, and consistency while retaining the structure and style of the specification, i.e. they should have a coherent organization, are not redundant, and are expressed separately.
- traceable: a requirement is traceable if its origin is clear (backward traceability) and it is referenced in later development steps (forward traceability).

The list gives an overview of quality criteria for requirements that have to be considered for a successful implementation of new systems. The standardization tries to support the successful development of SoS trough a template, the so-called use case methodology (cf. Section 3.3.4). This template supports the development of well-structured requirements. However, due to its extensive description of general and technical aspects, a manual check of all these criteria is difficult. Thus, aim of this PhD project is to support the syntactically correct and consistent acquisition and check of requirements for SoS. This shall be done automatically or rather semi-automatic.

3.3.3 Consistency Checks

As mentioned above, requirements are consistent if no conflicts within one requirement and between requirements exist (IEEE, 1998), i.e. requirements have to be characteristic, logical, and temporal consistent. Characteristic inconsistencies are conflicting objects, e.g. a requirement says the transfer rate of the traffic camera amounts 98.304.000 Bit/s, and another requirement implies that the data of the motion detector have to be analyzed. These both requirements assume two different techniques to record the current traffic volume, thus, an inconsistency between the applied object exist. Logical and temporal inconsistencies result from conflicts between described actions in requirements. For example, one requirement describes that the data have be to transferred and analyzed, another requirement implies that the data is analyzed and than transferred, i.e.two requirements describe a complete different behavior.

Inconsistencies within requirements written in natural or semi-formal language can be avoided with different known techniques, e.g. proofreading by a third-party (Rupp, C. and die SOPHISTen, 2014), glossaries (IEEE, 1998; Rupp, C. and die SOPHISTen, 2014; Robertson and Robertson, 2012), and text mining (Berry and Castellanos, 2007). Proofreading is the simplest one and is supported by various word processing software, such as MS Word or browser add-ons like spell checker (Smart Software, 2016). However, these tools cannot check whether the described requirements consistent or not; therefore, a human has to check the requirements. Creating a glossary has to be done manually; however, its consistent use can be tool-supported. For example, the word *lights* have different meanings, in the smart city context lights can be traffic lights or street lights, hence, a clear definition in a glossary is needed. Text mining uses linguistic analyses for extracting information and detecting text patterns; therefore, techniques like clustering, classification, and retrieval can be used (Berry and Castellanos, 2007). This approach can be applied automatically on requirements.

The use of text mining techniques for consistency checks operates with similarity checks. For detecting inconsistencies between requirements, requirements have to be found that have a high similarity because

inconsistencies occur frequently in requirements that touch the same areas of a system (Port et al., 2011). The process of text mining can be divided into two phases: text preparation and text analysis (Tan, 1999). The preparation adapts the text before it is analyzed to reduce its complexity. These include, for example, tokenization, stemming, and tagging (Kibble, 2013). Tokenization splits the text into tokens that can be single words or meaningful expression, e.g. *traffic flow* and *street lights*. Stemming removes all affixes, e.g. *supporting* is shorten into *support*. Tagging associates words with a grammatical category, e.g. *flow* with the type *noun*. A further text preparations are clustering techniques to group a set of data, so that similar subjects are grouped together in a particular cluster while dissimilar subjects are located in different clusters (Tan, 1999). Based on the preparation, a Latent Semantic Analysis (LSA) can be made that extracts and represents the contextual-usage meaning of words by statistical computations applied to a large corpus of text (Port et al., 2011). With regard to the results of the LSA, hints for inconsistencies can be given.

Aim of the PhD project is to adapt these techniques to the use case methodology, which is described in the next part.

3.3.4 IEC 62559-2: Use Case Methodology

Use Cases are described by the Object Management Group (OMG)(OMG, 2011) as follows: *"Use cases are a means for specifying required usages of a system. Typically, they are used to capture the requirements of a system, that is, what a system is supposed to do."*. They are an effective methodology to describe systems functionality and their boundaries (Cockburn, 2000; Rupp, C. and die SOPHISTen, 2014). The use case methodology is an accepted method to describe requirements in a structured way by natural language. Due to the presented template, editors of the use case do not forget information, e.g. the detailed description of actors and non-functional requirements. However, the description of any possibility and all boundaries of a system leads sometimes to inconsistencies within the use case description.

This methodology is already used by the standardization organization and an own template was developed for describing future SoS complete and consistent (DKE, 2014; IEC, 2015). Regarding the complexity of use case descriptions, a manual consistency check is difficult and should be tool-supported. A web-based application to describe and collect use cases already exist, the Use Case Management Repository (UCMR). It supports the development process by various libraries, but it does not have any possibility for an automated consistency check. The IEC use case template is introduced below to get a first impression of the methodology.

The use case template is structured in eight parts: description of the use case, diagrams of use case, technical details, step by step analysis, information exchanged, requirements, common terms and definition, and custom information. However for explaining the template in brief, it can also be split in four segments: general information, functional description, technical details, and additional information; that are subdivided in further parts. The use case description starts with general information, they may include an unique name, objectives and a boundary to other use cases or scenarios. On this basis, a complete description of the required functionality is given as continuous text. In the next step, the functional description is constituted as step-by-step analysis for different scenarios whereas information objects and further requirements are defined. In parallel with these three activities, terms are defined and additional information are given for supporting the understanding of the use case.

Figure 3 shows some parts of the use case template based on the example *Supporting the traffic flow in cities*. The tables *Name of use case* and *Scope and objectives of use case* are part of the first segment, general information on the use case are given. The unique identifier is *SC-01* and the use case is part of the domains *Mobility and Transport* as well as *Civil Security* and the zone *Process* (based on the Smart City Infrastructure Architecture Model (SCIAM)). In addition, a comprehensive name should be given for the use case. The second table demonstrates boundaries, limits, and aims of the use case as well as the link to city business cases. Thus, cities are defined for which the use case is relevant as well as objectives that should be reached through the use case, e.g. city buses on schedule and less traffic accidents. Also, relevant business cases are mentioned e.g. the future city planning. The third table lists non-functional requirements like the private and data protection laws that have to be considered by the implementation of the use case.

3.4 Related Work

In this part, an overview of related work regarding the topics requirements specification for smart spaces and consistency checks are given. Various approaches are considered and described in brief.

The approach by Evans et al. (Evans et al., 2014) describes a model to gather requirements for special systems, in their case an ambient assisted living system is considered. They develop a six-step activity list to collect and describe requirements in

1.1 Name of use case

Use case identification		
ID	***Area Domain(s)/ Zone(s)***	***Name of use case***
SC-01	Mobility and Transport, Civil Security / Process	Supporting the traffic flow in cities

...

1.3 Scope and objectives of use case

Scope and objectives of use case	
Scope	The traffic flow on main streets within cities that have a population more than 50,000 shall be accelerated, thus street buses keep their schedule.
Objective(s)	• intelligent switching of traffic lights • city buses on schedule • less traffic accidents
Related business case(s)	• Future city planning

...

6 Requirements

Requirements		
Categories ID	***Category name for requirements***	***Category description***
D	Data protection	The usage of traffic cameras has a great influence on the private and data protection laws.
Requirement ID	***Requirement name***	***Requirement description***
D-01	Storage of videos	The videos have to be deleted after analyzing the traffic volume, but not later than 10 minutes..
D-02	Disguise number plates	All number plates have to be unrecognizable.

...

Figure 3: Use Case Methodology - Example.

a scenario-based process. These activities are: establish high-level objectives, establish scope, identify stakeholders, identify tasks / functionalities, identify system performance quality, and determine stakeholder profiles. Furthermore, the process is complemented with two further activities for supporting operations and harmonizing requirements parallel to the mentioned activities. At first glance, it seams similar to the use case methodology (cf. Section 3). Both techniques describe objectives and scopes at the beginning for a common understanding of requirements (or rather use cases). The identification of stakeholders or rather actors follows the objectives and scopes. Afterwards, scenarios are identified in which stakeholders or actors are involved. Regarding the stakeholders or actors further requirements, almost non-functional requirements, are identified and linked with the described functional requirement or use case. These approaches differ in their given structure, the use case methodology seams more structured and detailed as the scenario-based approach by Evans et al. (Evans et al., 2014). Sutcliffe et al. (Sutcliffe et al., 2006) also describes a similar scenario-based approach. They describe their approach as an iterative process whereby the requirements specification is followed by mock-ups and prototypes to involve all stakeholders from the beginning in the specification process. Thus, the requirements description is more user oriented and scenarios regarding special aspects, such as cultural and local ones, are created.

The requirements engineering process for cyber-physical systems has to note mechanical engineering, electrical engineering, and computer science, i.e. all relevant aspects for smart city systems are considered (Wiesner et al., 2015). The approach by Wiesner et al. considers the requirements engineering process for future cyber-physical systems with a high number of components as a dynamic process that has to be supported by a universal content model and natural language processing technologies for collecting requirements. Therefore, domain specific models for various components and systems as well as their interfaces have to be implemented for using such models and technologies.

If methodologies and technologies reducing the amount of inconsistent requirements are considered, then approaches handling with inconsistent requirements can be also viewed to get another viewpoint on it. Some authors are convinced that inconsistencies within requirements can be tolerated when disposal costs are higher than savings for a better understanding (cf. (Apfelbacher, 2007) and (Parnas and Clements, 1986)). For these approaches, inconsistencies have to be classified and detected in requirements' specifications to make a decision on their relevance and expected disposal costs. The PhD thesis by Apfelbacher (Apfelbacher, 2007) describes an approach how inconsistencies within requirements, concepts, and models for software systems can be detected and how inconsistencies can be avoided from beginning. First, the classification of inconsistencies is demonstrated by four inconsistent classes: *unnoticed*, *semantical*, *specification-*, and *model-based* inconsistencies. These inconsistencies except the unnoticed one can be detected by different techniques partly with tool support, while a human have to decide whether it is an inconsistency or not. Apfelbacher says that more than one reviser have to check the requirements specification to be sure that important inconsistencies were removed. The reviser, in the best

case, are some developer and receiver of the requirements, so that both viewpoints of the requirements specification process are considered. This is particularly important for checking the model-based inconsistencies because these are often part of the conceptual product description to be established. This proofreading concept can be support by tools, such as UML tools or word processors, to facilitate and accelerate the consistency check especially for semantically inconsistencies. Additionally, templates for the consistency check can be created which contain e.g. checklists to remember proofreader on important things e.g. special specification guidelines.

To avoid inconsistencies from beginning, Apfelbacher describes a procedure to create requirements and conceptual documents. His suggestion is a structured procedure which based on a good communication between all participants. It starts with the determination of the general procedure of the documentation process and the classification of possible inconsistencies so that each project has to decide which parts are relevant for them. Afterwards, a visual description of requirements and a concept, e.g. via UML diagrams, are suggested which should be extended by notes to improve the understanding. Finally, notes and diagrams should be prepared, and these documents should be proofread similar to the procedure for consistency checks.

The described procedure of creating structured text and the hint to use tool support are two aspects which are also considered in this work. However, the aim is to remove inconsistencies, if they are detected by a reviser whereas it should his or her decision what is done with the inconsistency. Though, the classification of inconsistencies can be used for detecting and rating inconsistencies for the reviser.

4 METHODOLOGY

The PhD project is structured by a template for design science research. *Design science research* is a pragmatic research paradigm to solve real-world problems through the usage of information technology (IT) artifacts with a high priority on relevance in the application domain (Hevner and Chatterjee, 2010). Thus, the design science research links information systems research and practice. For example, a real world problem is the collection of consistent requirements for systems particularly for complex systems, such as smart cities (DKE, 2014). These systems are SoS, i.e. each subsystem has its own requirements list, and additionally, functional requirements for the complex system and interface requirements are needed to depict a requirements list for a SoS. The *Design Science Research Process (DSRP)* model by Peffers et al. (Peffers et al., 2006) is used to consider this real world problem. The DSRP model describes a conceptual model for researchers to carry out design science research for information systems and requirements engineering (Peffers et al., 2006). It provides a template for readers and reviewers to recognize and evaluate research work which is used for the PhD project (Peffers et al., 2006). The template for the DSRP model is shown in Figure 4.

The first step of the DSRP model is the *problem identification and motivation* which defines a problem and should show the importance of a problem, for motivating researchers to pursue a solution and to accept results. Additionally, it helps to understand the thinking on the problem. At the end of this step, the state of the problem and the importance of its solution are clear. The second step is *objectives of a solution*. In this step, objectives are inferred from the problem specification. These objectives can be quantitative or qualitative, i.e. indications that can be used to compare the solution with the current state or whether a new artifact is expected to support the solution. Therefore, knowledge about the problem specification and current solutions and their efficacy are needed. The third step consists of *design & development*, at this point the chosen solution have to be implemented. Next to the realization of the solution, this step includes the description of requirements and the resultant architecture of a solution. Therefore, theory knowledge is needed to create a solution. The fourth step is the *demonstration*. The solution is applied e.g. in experimentations, simulations, case studies, etc. to show its functionality and efficiency. The fifth step is the *evaluation*. The demonstration of the solution is observed and metrics and analyses are used to evaluate the solution. Thereby, the effectiveness and efficiency are considered. If results of the observation do not meet the expected results, the process will iterate back to the second or third step to adapt the expected objectives and implementation as well as to evaluate the results again (also called *Design Process*). Otherwise, the process ends with the sixth step, the *Communication*; however, this step can be reached from each step to share ideas, results, and new findings (Peffers et al., 2006).

Based on the above mentioned process, this work is structured by the DSRP model. During the first step, difficulties in the standardization process for SoS were observed in standardization workgroups and read in various reports (DKE, 2014; DKE, 2015). These discoveries and the application of software engineering techniques, such as the use case method-

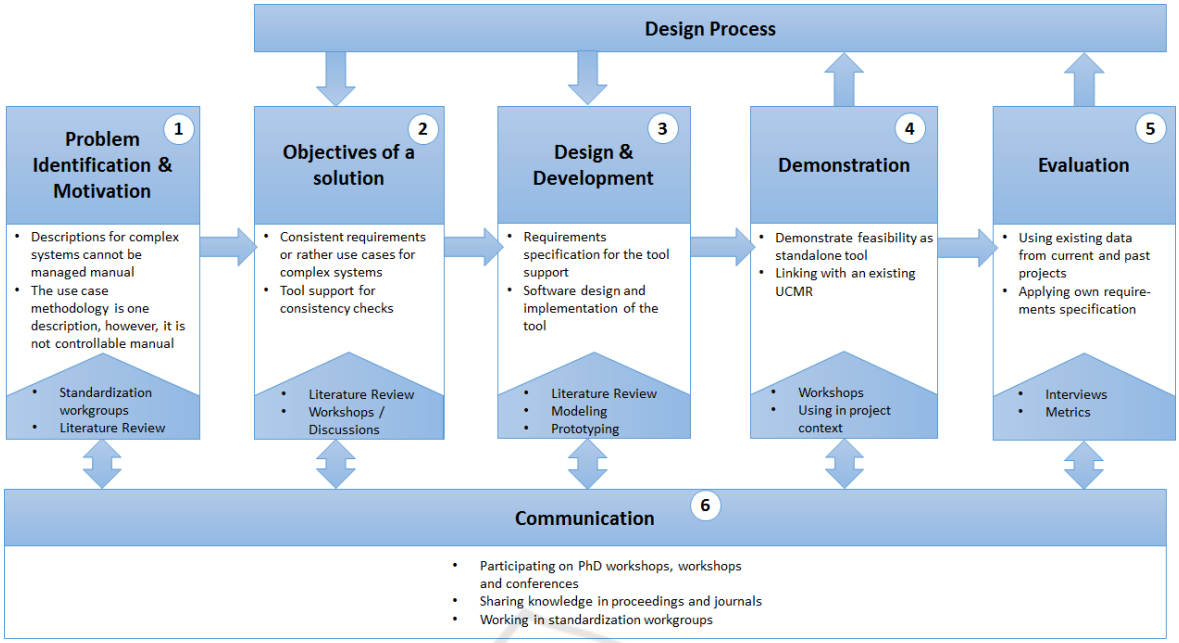

Figure 4: Design Science Research Process Model (according to (Peffers et al., 2006)).

ology (cf. Section 3), led to the a research question for supporting the collection and management of requirements (cf. Section 1). Syntactical correct and consistent requirements can support the requirements specification or rather the use case methodology, thus, these both properties and existing approaches are considered in this work. After creating an overview of existing techniques and approaches, the design & development process starts with a requirements specification and follows with an implementation. It should also be pointed out that literature is reviewed and models as well as prototypes are implemented. The demonstration of the tool should be done as standalone tool for checking requirements that are not created as use cases, e.g. the requirements specification of the tool itself, and as link to a UCMR. For the evaluation metrics are used to measure the amount of detected incorrect and inconsistencies after executing a consistency check. In addition, interviews with users of the UCMR are carried out. In parallel with these five steps, ideas and results are presented on workshops, proceedings, and standardization workgroups.

5 EXPECTED OUTCOME

The use case methodology (cf. Section 3) shall be extended by an application for consistency checks. This application shall be implemented for the UCMR; however, the extension shall also be an independent application for executing consistency checks for requirements lists that do not base on the use case approach. Figure 5 shows the concept of the UCMR and an included consistency check. The UCMR is a web-based application for creating, collecting, and managing use cases by different experts. Additionally, it allows an import of use cases as XML or HTML files, as well as an export. The consistency check shall be an external application that is integrated within the UCMR to provide a final check before use cases are stored or rather shared with other experts. The consistency check shall also create a report, which can be exported by the experts as HTML file.

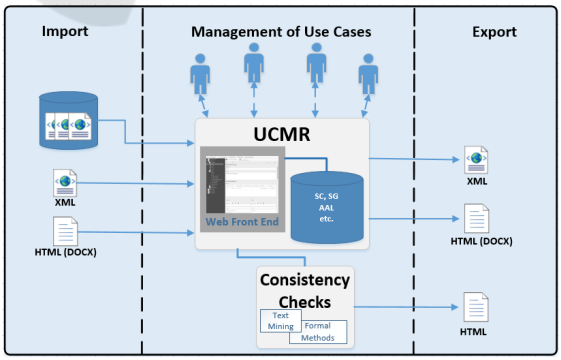

Figure 5: Design.

Results of this work assist customer or experts who create requirements (e.g. various experts, such as ICT and domain experts for smart homes, smart grids, smart cities, etc.), requirements engineers who manage requirements, and system architects who work

with these requirements. Next to the requirements improved, the development of new standards that base on requirements for future systems can be supported; thus, standardization organizations are further beneficiaries. Additionally, all parties should be supported through visualization techniques during the requirements specification process to create an overview of existing requirements and their relations, maybe it can support the completeness of requirement specifications.

6 STAGE OF THE RESEARCH

Currently, I am reviewing literature for existing techniques to reduce syntactically incorrect and inconsistent requirements. Concurrently, techniques for visualizing requirements are considered. Based on the literature review, requirements for the consistency check applications are collected and recorded. Regarding on the shown DSRP model, the work is at the second step *objectives of a solution*, and regarding on the defined objectives in Section 2, the stage of research is as follows:

- visualizing requirements: The literature review for visualizing techniques is at the beginning; thus, papers that give an overview of visualizing requirements were read (Khairuddin and Hashim, 2008; Heim et al., 2008; Lamsweerde, 2009).
- reducing syntactically incorrect requirements: The literature review for techniques to avoid syntactically incorrect requirements is also at the beginning, difficulties are described in (IEEE, 1998).
- reducing inconsistent requirements: To get consistent requirements various techniques for the different kind of requirement descriptions (natural language, semi-formal method, and formal method) already exist. Some of these techniques were considered and described on the survey, e.g. simple proofreading (Rupp, C. and die SOPHISTen, 2014), using a glossary (IEEE, 1998; Rupp, C. and die SOPHISTen, 2014; Ludewig, 2007), Text Mining (Berry and Castellanos, 2007), Software Cost Reduction method (Heitmeyer et al., 1996), etc.
- implementing an approach: Based on the literature review a requirements list, design, and implementation are created and linked with the UCMR.
- evaluating the approach: First validation data were collected. The first validation shall be done with the describe requirements for the consistency check application (self-review). Further validations base on use cases from the DISCERN project (Discern, 2016) and various results from German standardization organizations.

REFERENCES

Apfelbacher, R. (2007). *Tolerierbare Inkonsistenzen in Konzeptbeschreibungen*. PhD thesis, University of Postdam.

Berry, M. W. and Castellanos, M., editors (2007). *Survey of Text Mining: Clustering, Classication, and Retrieval*, volume 2. Springer.

CISCO and AGT International (2014). AGT and Cisco Traffic Incident Management Solution: Improving Traffic Safety and Efficiency. Technical report, CISCO and AGT International.

Cockburn, A. (2000). *Wrting Effective Use Cases*. Addison-Wesley Professional.

DaCosta, F. (2013). *Rethinking the Internet of Things: a scalable approach to connecting everything*. Apress.

Discern (2016). Discern - Distributed Intelligence for Cost-effective and Reliable Solutions. http://www.discern.eu/ last visit on 28th January 2016.

DKE (2014). The German Standardization Roadmap Smart City. Technical Report 1, DKE.

DKE (2015). The German Standardization Roadmap Smart City. Technical Report 1.1, DKE.

Evans, C., Brodie, L., and Augusto, J. C. (2014). Requirements Engineering for Intelligent Environments. In *International Conference on Intelligent Environments*.

Heim, P., Lohmann, S., Lauenroth, K., and Ziegler, J. (2008). Graph-based Visualization of Requirements Relationships. In *Requirements Engineering Visualization*, pages 51–55. IEEE.

Heitmeyer, C., Jeffords, R., and Labaw, B. (1996). Automated consistency checking of requirements specications. In *ACM Transactions on Software Engineering and Methodology (TOSEM)*.

Hevner, A. and Chatterjee, S. (2010). Design Research in Information Systems. In *Integrated Series in Information Systems 22*, pages 9–21. Springer Science+Business Media.

IEC (2015). IEC 62559-2:2015 Use case methodology - Part 2: Definition of the templates for use cases, actor list and requirements list.

IEEE (1990). Standard: IEEE Std 610 - IEEE Standard Glossary of Software Engineering Terminology.

IEEE (1998). IEEE Recommended Practice for Software Requirements Specification.

ITU-T (2012). Overview of the Internet of things. Technical Report Y.2060, ITU.

ITU-T FG-SSC (2014). Smart sustainable cities: An analysis of definitions . Technical report, ITU-T.

Johnson, C. (2011). *Achieving Systems Safety*, chapter CyberSafety: On the Interactions between CyberSecurity

and the Software Engineering of Safety-Critical Systems, pages 85–95. Springer.

Khairuddin, N. N. and Hashim, K. (2008). Requirements Visualization Techniques: A Comparative Analysis. In *Proceedings of the 8th Conference on Applied Computer Science, ACS08*.

Kibble, R. (2013). Introduction to natural language processing. Technical report, University of London.

Lamsweerde, A. v. (2009). *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley.

Ludewig, J. (2007). *Software Engineering*. dpunkt Verlag.

OMG (2011). Unified Modeling Language. Technical report, OMG.

Parnas, D. L. and Clements, P. C. (1986). A Rational Design Process: How and Why to Fake It. *IEEE Transactions on Software Engineering*, (2):251–257.

Peffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V., and Bragge, J. (2006). The Design Science Research Process: A Model for Producing and Presenting Information System Research. In *DESRIST*, pages 83–106.

Port, D., Nikora, A., Hayes, J. H., and Huang, L. (2011). Text Mining Support for Software Requirements: Traceability Assurance. In *System Sciences (HICSS)*.

Rittel, H. W. and Webber, M. M. (1973). Dilemmas in a general theory of planning. *Policy sciences*.

Robertson, S. and Robertson, J. (2012). *Mastering the Requirements Process - Getting Requirements Right*. Addison-Wesley.

Rupp, C. and die SOPHISTen, editor (2014). *Requirements-Engineering und -Management*. Hanser Verlag.

Smart Software (2016). Spell Checker for Chrome.

Sommerville, I. (2009). *Software Engineering*, volume 9, chapter Formal Specification. Addison Wesley.

Sommerville, I., Cliff, D., Calinescu, R., Keen, J., Kelly, T., Kwiatkowska, M., McDermid, J., and Paige, R. (2011). Large-scale complex IT systems. *Communications of the ACM*, 55(7):71–77.

Sutcliffe, A., Fickas, S., and Sohlber, M. M. (2006). PC-RE: a method for personal and contextual requirements engineering with some experience. *Requirements Eng*.

Systems and Software Engineering Directorate (2008). Systems Engineering Guide for Systems of Systems. Technical report, Office to the Deputy Under Secretary of Defense for Acquisition and Technology, Washington, DC.

Tan, A.-H. (1999). Text Mining: Promises and Challenges. In *South East Asia Regional Computer Confederation*.

Tragos, E. Z., Angelakis, V., Fragkiadakis, A., Gundlegard, D., Nechifor, C., Oikonomou, G., Pöhls, H. C., and Gavras, A. (2014). Enabling Reliable and Secure IoT-based Smart City Applications. In *The First Internation Workshop on Pervasive Systems for Smart Cities*.

Wagner, S. (2014). Verkehrssteuerung über die Siemens private cloud". Technical report, Siemens.

Wiesner, S., Baalsrud Hauge, J., and Thoben, K.-D. (2015). Challenges for Requirements Engineering of Cyber-Physical Systems in Distributed Environments. In *Advances in Production Management Systems: Innovative Production Management Towards Sustainable Growth*.