Engineering Trust- and Reputation-based Security Controls for Future Internet Systems

Kristian Beckers, Maritta Heisel,
Francisco Moyano and Carmen Fernandez-Gago

September 1, 2015

Abstract

Reputation as a decision criteria for whom to trust has been successfully adopted by a few internet-based businesses such as ebay or Amazon. Moreover, trust evaluation is becoming of increasing importance for future internet systems such as smart grids, because these contain potentially millions of users, their data, and a huge number of subsystems. The resulting scale and complexity makes them ideal candidates for trust and reputation based security controls, but currently engineering methodologies are missing that provide structured step-by-step instructions on how to design such controls. We contribute such a methodology including tool support that helps (i) to elicit trust relationships, (ii) to reason about how to construct trust and reputation engines for these and finally (iii) to specify consequent security controls. The methodology is based on formal OCL-expressions that provide (semi-)automatic support analysing UML models with regard to trust and reputation information.

1 Introduction

Future Internet Systems introduce new ways of communication among software-based devices. For example, smart grids are commodity networks that intelligently manage the behaviour and actions of its participants, where in our running example the relevant commodity is electricity. Smart grid is envisioned to be a more economic and sustainable supply of energy, and an essential feature of smart grids is that there is two-way electronic communication between energy providers and consumers. It is envisioned that consumers can use their individual devices such as smart phones, notebooks, etc. to communicate with their energy providers, control the energy consumptions of their homes and use further services. The envisaged scale of smart grids is enormous if we consider that in Europe alone live around 500 million people[1] and most of them will use different devices to connect to the grid. Hence, the security challenges as well as new vulnerabilities will raise in future internet systems[2].

[1] http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=demo_pjan&lang=en

[2] http://www.nessos-project.eu/media/deliverables/y3/NESSoS-D4.3-PartII-Roadmap.pdf

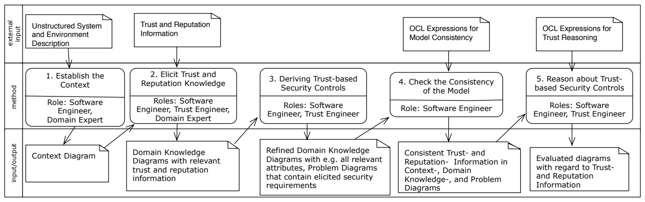
external input
method
input/output
Unstructured System and Environment Description
Trust and Reputation Information
OCL Expressions for Model Consistency
OCL Expressions for Trust Reasoning
1. Establish the Context
Role: Software Engineer, Domain Expert
2. Elicit Trust and Reputation Knowledge
Roles: Software Engineer, Trust Engineer, Domain Expert
3. Deriving Trust-based Security Controls
Roles: Software Engineer, Trust Engineer
4. Check the Consistency of the Model
Role: Software Engineer
5. Reason about Trust-based Security Controls
Roles: Software Engineer, Trust Engineer
Context Diagram
Domain Knowledge Diagrams with relevant trust and reputation information
Refined Domain Knowledge Diagrams with e.g. all relevant attributes, Problem Diagrams that contain elicited security requirements
Consistent Trust- and Reputation- Information in Context-, Domain Knowledge-, and Problem Diagrams
Evaluated diagrams with regard to Trust- and Reputation Information

Figure 1: Our Method for Engineering Trust- and Reputation-based Security Controls for Future Internet Systems

The concept of trust has been in discussion for a long time and researchers still work on clarifying its terminology [17]. In addition, several well known applications rely on trust and reputation mechanisms such as Amazon's product ratings and ebay's seller feedback [14].

In security engineering, current practice is to mitigate potential threats by using hard trust approaches. These approaches are based on cryptography and strict rules to prevent access to resources without proper authorisation. These mechanisms often entail a high computational power or administrative burden, they are hard to maintain in dynamic environments [20], and they only provide limited control prior to the access of users; once users are in the system, hard trust approaches do not focus on detecting abnormal behaviours. Moreover, any misbehaviour may lead to multiple rule updates, which can lead to missing rules due to limited IT staff or to wrong rules due to human errors.

In contrast to hard trust mechanisms, soft trust mechanisms rely on social control and on the characterisation of trust relationships based on certain factors that influence these relationships [19]. Examples of these factors are previous experience, membership to a group, or reputation. Trust values can be used by the trustor itself (i.e. the entity placing trust) to evaluate if it should engage in an interaction with other entities. The main difference with the previous schemes is that we are empowering entities to make decisions based on personal judgement of its context and individual knowledge. Trust is no longer based on a set of strict rules or on a statement by a certification authority that is trusted by definition. In particular, future internet systems require not to depend on a set of strict rules for security, because the changes to these rules would result in overwhelming effort when coping with the scale and evolution of these systems. Although trust and reputation have been barely considered in requirements engineering, some works have included some of their concepts as part of this phase [18][21].

We propose trust- and reputation-based security controls for a smart grid in our previous work [16]. We rely on the problem frame approach [13] for this analysis, because in contrast to goal-based methods such as SI* [15] and KAOS [23], problem frames use abstractions of the system-to-be, emphasizing the context that surrounds the system, which is particularly important in order to analyse trust relationships and reputation. The interested reader can find a more detailed discussion on our choice in our previous work [16].

This work presents two main extensions over the previous work. First, we discuss a detailed step-by-step methodology that allows for designing trust-based security controls for future internet systems. This methodology supports the separation of duties among the domain expert, the software engineer and the trust engineer. Second, we specify formal checks for UML4PF (a UML profile and tool support, see Sect. 2 for details) models using OCL [22]. These checks support consistency of different diagrams in one model, help eliciting trust relationships and reasoning about the design of trust and reputation engines. Finally, we provide tool support for creating trust models using UML4PF and (semi-) automatically applying the OCL checks on these models.

These contributions empower ordinary software- and security-engineers to apply the skills of engineering trust-based security controls.

The remainder of the paper is structured as follows. Section 2 shows the UML4PF framework, and Sect. 3 presents our step-by-step methodology for engineering trust-based security controls. Section 4 illustrates the application of our methodology and Sect. 5 discusses our results. Section 6 concludes and gives directions for future research.

2 The UML4PF Framework

The UML4PF framework[3] is a result of a decade of research of the University of Duisburg-Essen in collaboration with the ITESYS GmbH. Several projects contributed to the project such as the EU project Network of Excellence on Engineering Secure Future Internet Software Services and Systems (NESSoS), and the project of the German State of North Rhine-Westphalia and EFRE ClouDAT, and the DFG project GenEDA. The practitioners of the ITESYS GmbH and the ones in the projects ensure that the resulting framework, its tools, and methods meet practitioners' needs. In addition, the developed framework is available as open-source, which allows interested parties to try and use the framework for free.

We use the UML representation of the problem frames method called *UML4PF* [9], because this allows us to write OCL expressions to validate the models that will be included in the UML4PF support tool. Moreover, we aim to integrate this analysis into a structured software development process, e.g., an extension of the ADIT [8] process that relies on UML4PF. We choose the UML notation, because software engineers are familiar with it to express software design choices. Moreover, if we express the software analysis and design in UML, we do not need to map the analysis results to a different notation for the software design. This reduces one source of mistakes during software development. Hence, expressing trust and reputation analysis in UML allows for a seamless refinement step to software design, by re-using the UML models created during the analysis phase in the software design phase.

UML4PF provides a framework for step-by-step methodologies to achieve different quality attributes of a system-to-be. To this date UML4PF supports the following methodologies:

- A software development process with over 70 traceability and consistency checks between different diagrams [8].

[3]`http://www.uml4pf.org`

Table 1: OCL Expressions that support Trust-based Security Reasoning and Consistency Checks

OCL-EXPR-ID	**Referenced Class**	**Expression**	**Supporting Analysis Questions**
Reasoning Support			
IDHE001	HumanEntity	- List all biddable domains that are not a human entity	- Are human entities missing?
IDEN001	Entity, HumanEntity	- List all domains that are not entities or human entities	- Are some entities or human entities not elicited yet?
IDHE002	HumanEntity	- List all human entities that do not have a trusts relation.	- Are trust relations of HumenEntities missing?
IDEN002	Entity	- List all entities that do not have a trusts relation.	- Are trust relations of Entities missing?
TRTE001	TrustEngine	- List all machine domains that have a direct relation to a TrustEngine	- Are Trust Engines missing in the model?
TRRE001	ReputationEngine	- List all TrustEngines that have a direct relation to a ReputationEngine	- Are ReputationEngines missing in the model?
TRJE001	TrustFactor	- Check that the how attribute is set and it is set either to ”assigned” or ”monitored”.	- Are all the trust factors either “assigned” or “monitored”?
TRCL001	Claims	- Check that claims have set the when attribute to either to “after interaction” or “any moment”.	- Do all claims specify when they must be provided.
TROF001	ObjectiveFactor	- List all trust relationships that have no objective factors.	- Are all objective factors of the entity considered?
TRSF001	SubjectiveFactor	- List all trust relationships that have no subjective factors.	- Are all subjective factors of the entity considered?
Consistency Checks			
CCRS001	HumanEntity	- Check that all HumanEntities have the value trustRole set.	- Are HumanEntities modelled correctly ?
CCRS002	Entity	- Check that all EntitiesEntities have the value trustRole set.	- Are Entities modelled correctly ?
CCCL001	Claim	- Check that all sources of claims are a Human Entity	- Are claims modelled correctly ?
CCCL002	Claim	- Check that all targets of claims are an Entity or Human Entity	- Are claims modelled correctly with respect to entities?
CCCL003	Claim	- Check that all claims have targets and sources	- Have claims an origin and a target ?
CCSF001	SubjectiveFactor	- Have subjective factors the who value set and refer to a trust relationship?	- Are trust relationships modelled considered subjective factors ?
CCTV001	TrustValue	- Check that all dependencies with a trusts relationship have a dependency to a TrustValue	- Are trust relationships modelled completely ?
CCLTR001	Trustor, Trustee	- Check that trust relationships have a trustor and a trustee.	Are all trust relationships modelled correctly?
CCLTR002	Trustfactor	- Check that All classes with a stereotype trustfactor including the inheriting classes subjective and objective factor have a dependency to a trusts relationship or to an entity	Are all trust factors refer to trust relationship or to entities?

- A dependability profile and tool support [11].
- A computer aided threat analysis based on functional requirements [10].

- A support methodology for security analysis [3, 6] compliant to the Common Criteria standard [12].
- A support methodology for early hazard analysis complaint to the ISO 26262 automotive safety standard [5].
- A method for modelling of variability by using feature modelling for the problem and the solution space [2].
- A methodology [1] to identify unwanted interactions between requirements.
- A privacy threat analysis method [4] based on information flow in requirements models.

In this paper, we rely on all the extensions introduced so far and provide support for engineering trust-based security controls. Our methodology shows step-by-step how to identify trust relationships, refine these relationships and elicit trust values, consider reputation of entities participating in the trust relationships, and creating an analysis for evaluating the security level of an asset. We provide a dedicated homepage for the UML4PF Trust Extension[4] that contains our previous work, the support tool and a technical report of our current work.

3 Engineering Trust-based Security Controls for Future Internet Systems

We show our method (see Fig. 1) for engineering trust-based security controls in the following. This method requires three roles to collaborate: a domain expert, a software engineer, and a security engineer familiar with trust engineering that we call Trust Engineer.

Step 1: Establish the Context Trust relations are only valid for a specific context. The software engineer and the domain expert describe the context of the software development in a context diagram. This diagram describes the machine, the thing to be built, in its environment using domains and interfaces between these domains. A set of textual functional requirements refers to the domains in the context diagram. Afterwards, the security engineer elicits assets and security requirements for them.

Step 2: Elicit Trust and Reputation Knowledge The domain expert and the trust expert have to work together. The former elicits trust-unaware domain knowledge diagrams, whereas the latter (together with the former) provides an initial *trust domain knowledge*, where the high level aspects of the trust and reputation models are first sketched. These aspects include specifying trust entities, their trust relationships, claims, and trust factors. This step is supported by the OCL expressions in Tab. 1, which help to identify incomplete relationships and missing relationship elements such as entities.

[4]`http://www.uml4pf.org/ext-trust/`

Step 3: Deriving Trust-based Security Controls The information in the trust domain knowledge diagrams is refined in this step by the software and trust engineers. The final diagrams contain detailed information about trust and reputation relationship, e.g. roles played by the entities, insight on the claims, the trust values, and objective and subjective factors[5]. In addition, the security engineer and the software engineer collaborate to analyse how the respective trust and reputation engines integrate into the system-to-be and their relationship with the system requirements and the machine.

Step 4: Check the Consistency of the Model The work with any kind of models is subject to the risk that engineers make mistakes while creating them. We identified several common mistakes that are made when using our UML4PF trust extension. We defined OCL expressions to find several of these in the *Consistency Checks* part of the OCL expressions in Tab. 1. These contain boolean checks for the conditions in the table. The execution of these checks support software engineers to find these common mistakes without having to consult the trust engineers.

Step 5: Reason about Trust-based Security Controls Besides mistakes in models that can be identified clearly, there are other issues that require a detailed discussion among engineers. For example, issues regarding the completeness of information of a trust relationship. Support for these reasoning based concerns cannot fully be automated. However, we provide OCL expressions in the *Reasoning Support* part of Tab. 1 that search for specific relations between model elements that may provide hints for conceptual problems in the model. The trust engineers and the software engineers consider the resulting elements in an analysis and use them as a basis for discussion and finally to improve their diagrams.

4 Application of our Method

We use the Common Criteria protection profile for the smart metering gateway [7] as an example for our approach. The protection profile defines security requirements for a smart metering gateway. In addition, we use the trust and reputation extension of UML4PF's UML profile introduced in our previous work [16].

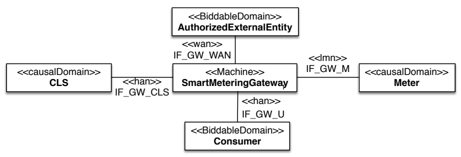

Figure 2: Smart Metering Gateway (Context Diagram)

[5]For further information on the semantics of the trust and reputation concepts, we refer the reader to our previous work [16].

Step 1: Establish the Context UML4PF describes a system in domains. The class with the stereotype *machine* represents the thing to be developed (e.g., the software), and *CausalDomain*s comply with some physical laws, while *BiddableDomain*s are usually people. Domains are connected by interfaces consisting of *shared phenomena*. Shared phenomena may be events, operation calls, messages, and the like. They are observable by at least two domains, but controlled by only one domain, as indicated by an exclamation mark. For simplicity's sake, we show only the most relevant phenomena in our diagrams.

The gateway is a part of the smart grid and enables the two-way communication between energy providers and consumers. Moreover, smart metering systems meter the production or consumption of energy and forward the data to external entities. This data can be used for billing and steering the energy production.

Figure 2 shows the context diagram that describes the machine to be built in its environment. The ≪Machine≫ is the SmartMeteringGateway, which serves as a bridge between the Wide Area Network ≪wan≫ and the Home Area Network ≪han≫ of the Consumer. The Meter is connected to the machine via a Local Metrological Network ≪lmn≫. This is an in-house equipment that can be used for energy management. The Controllable Local System CLS can be, for example, a heater. The Meter sends meter data to the SmartMeteringGateway. The SmartMeteringGateway stores this data. The Meter can also receive updates from the AuthorizedExternalEntity forwarded via the SmartMeteringGateway. The AuthorizedExternalEntity receives meter data in fixed intervals from the SmartMeteringGateway. The Consumer can retrieve meter data via the SmartMeteringGateway. The Consumer can also configure the SmartMeteringGateway, send commands to the CLS, receive status messages from the SmartMeteringGateway and store user data in it.

Step 2: Elicit Trust and Reputation Knowledge Once the context is established, trust and reputation information must be elicited. We show in Fig. 3 a domain knowledge diagram focusing on the main elements of one trust relationship between the ≪HumanEntity≫ Consumer and the ≪Entity≫ CLS. The trust relationship has a ≪TrustValue≫ and there is a ≪SubjectiveFactor≫ associated to the Consumer.

On the other hand, Fig. 4 shows relevant information for reputation purposes. Concretely, we are specifying which entities can make ≪Claim≫s about others, and which objective factors are considered to yield those claims. In this example, an ≪HumanEntity≫ AuthorizedExternalEntity can make ≪Claim≫s about the ≪Entity≫ CLS, and the ≪ObjectiveFactor≫ UnplannedReparis refers to the CLS.

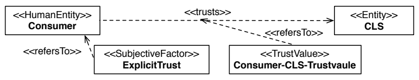

Figure 3: Analysing the Trust Relationship Consumer-CLS (Domain Knowledge Diagram)

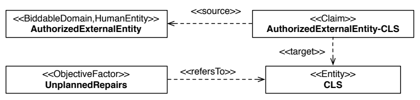

Figure 4: Analysing Reputation Information(Domain Knowledge Diagram)

Step 3: Deriving Trust-based Security Controls In this step, we derive trust-based security controls. For this purpose, the first task consists of refining the trust and reputation information in order to provide more insight about the trust and reputation models that are to be implemented. We achieve this by showing the attributes of all the trust and reputation elements, as depicted in Fig. 5.

The Consumer plays a *trustor* role in the Consumer-CLS-Trust relationship, it uses the subjective factor ExplicitTrust for this relationship and his trust disposition is neutral[6]. The ExplicitTrust subjective factor has 3 as initial value and is *assigned* by the Consumer. The Consumer-CLS-Trustvalue is an unidimensional continuous value between 0 and 5, with threshold value 3. This latter value refers to the threshold over which we assume that a trustor trusts a trustee. The CLS plays a *trustee* role in the Consumer-CLS-Trust relationship, although it also plays the *target* role with regard to the AuthorizedExternalEntity-CLS claim. It presents an objective factor, UnplannedRepairs, which is monitored (in contrast to manually assigned). The AuthorizedExternalEntity plays a *source* role because it can make claims about the CLS after an interaction with it. Claims are about the past behaviour of the target, and are represented by an unidimensional discrete number between 0 and 10. For deriving a fitting control, we need to specify trust and reputation engines as depicted in Fig. 6, where we also show the interactions between the ≪Machine≫, the ≪TrustEngine≫ and the ≪ReputationEngine≫. Both engines yield continuous values. The trust engine can retrieve reputation values from the reputation engine in order to compute trust values[7].

[6]We consider that for this scenario, a neutral trust disposition is reasonable, whereas other scenarios might require assuming that trust dispositions are lower or higher.

[7]This is the traditional way of relating trust and reputation: reputation is a valuable source of information for trust computation.

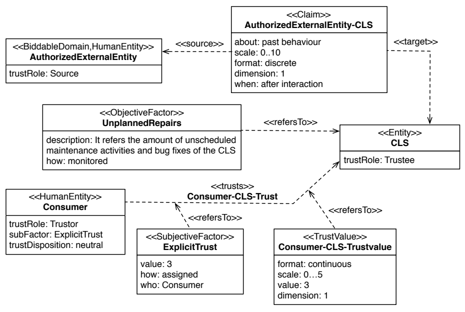

Figure 5: Refinement of Trust and Reputation Information (Domain Knowledge Diagram)

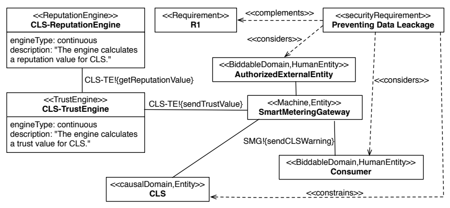

Figure 6: Describing Trust- and Reputation Engines(Problem Diagram)

The SmartMeeteringGateway can retrieve trust information and act accordingly. We focus our analysis on the *MeterData* as an asset. The meter data has value for the *Consumer*, because his/her billing depends upon it and a behaviour profile about the *Customer* can be created from it and it is of concern in the ≪Claim≫s illustrated in Fig. 3 and Fig. 4.

We consider the following functional requirements of the smart metering gateway in our example: **R 1** The CLS can receive energy consumption data from the Meter. We elicit the security requirement Prevent Data Leakage that ≪complements≫ the functional requirement R1. If the value of the Consumer-CLS-Trustvalue, which is computed by the trust engine, is above the minimum trust threshold (initially set to 3), no action shall be taken. In addition, the opinion in the form of feedback claims by the AuthorizedExternalEntity ≪AuthorizedExternalEntity-CLS≫ will be used by the reputation engine, which shall alert the Consumer about a possible problem.

Step 4: Check the Consistency of the Model We use the OCL expressions that are part of the consistency checks in Tab. 1 in this step. In particular, we illustrate the expression CCCL001 in detail in the following. The expression shown in listing 1 collects all classes with the stereotype ≪Claim≫ (lines 1-8) and it collects all dependencies with the stereotype ≪source≫ (lines 9-15). The expression filters the dependencies that start at a class with the stereotype ≪Claim≫ and end a class with the stereotype ≪HumanEntity≫ (lines 16-21). Finally, the expression subtracts the classes with the stereotype ≪Claim≫ that are at the end of the previously mentioned dependencies from all classes with the stereotype ≪Claim≫ (lines 22-24). In our case all the claims originate from the human entities and the expression returns an empty set. Otherwise we would get a list of classes for analysis.

Listing 1: CCL001. List all sources of claims that are not a Human Entity

```
 1 let stereotypeMain : String = 'Claim' in
 2 let claimClasses : Set (Class) =
 3   Class.allInstances()->select(
 4   let first : Set(Stereotype) = getAppliedStereotypes()->asSet() in
 5   first->union(first->closure(general.oclAsType(
 6   Stereotype))).name->includes(stereotypeMain)) in
 7 let stereotype : String = 'source' in
 8 let Sources : Set (Dependency) = Dependency.allInstances()->select(
 9         let first : Set(Stereotype) = getAppliedStereotypes()->asSet() in
10         first->union(first->closure(general.oclAsType(
11         Stereotype))).name->includes(stereotype)) in
12  let stereotypeSource : String = 'Claim' in
13  let stereotypeTarget : String = 'Human Entity' in
14  let DependencyClaims : Set (Dependency) =
  Sources->select(source.getAppliedStereotypes().name
      ->includes(stereotypeSource) and target.getAppliedStereotypes().name
      ->includes(stereotypeTarget)) in
 let correctClaims : Set (Class) =
  DependencyClaims.source.oclAsType(Class)->asSet() in
  claimClasses-correctClaims
```

Step 5: Reason about Trust-based Security Controls This step concerns the reasoning of the trust and reputation analysis. The engineers use the OCL expressions in support of reasoning (see Tab. 1). We illustrate the use of the expression TROF001 that lists all trust relations that have no objective trust factors. The result is that the Customer-CLS-Trust relationship (see Fig. 5) has no ≪Objective Factor≫s. Note that the ≪Objective Factor≫ UnplannedRepairs belongs to the ≪Entity≫ CLS and is part of a reputation ≪Claim≫ and not the trust relations. The discussions between the engineers fosters another possible ≪Objective Factor≫. It is the factor unauthorised data transmission of personal information from a CLS. The *who* value of the factor is the Consumer.

5 Discussion

We analysed our methodology for engineering trust- and reputation-based security controls based on the publicly available Common Criteria Protection Profile. We discussed the approach with the security partitioners in the ClouDAT project[8] in a brainstorming session. We presented the methodology to practitioners in the field of security engineering that were familiar with the Protection Profile. As a result, we found that our methodology helped the practitioners to distinguish between the concepts of trust and reputation. In particular, the practitioners mentioned that this structured procedure: Helps to identify trust relationships, supports the identification of reputation claims, helps to not forget relevant entities and their attributes, and supports the creation of consistent trust and reputation diagrams. Nevertheless, the practitioners raised the following concerns: The results of the OCL reasoning expression might lead engineers to add random elements to ”achieve” completeness, reading of the output of all expressions is time consuming, the UML profile and the methodology have to be learned beforehand, and our method does not integrate into common security development life cycles such as the Microsoft SDL.

6 Conclusions

We have presented a methodology to model and specify security controls based on trust and reputation. We pursue three main goals with this methodology; first, we aim to alleviate the security challenges that arise from highly dynamic, ever-changing, resource-constrained systems framed within the Future Internet, where soft security approaches become more adequate and flexible than traditional hard security ones; second, we consider the importance of a clear separation of duties, namely the domain expert, the

[8]`http://ti.uni-due.de/ti/clouddat/en/`

software engineer, and the security engineer. As systems grow in complexity, this separation becomes more advantageous and necessary. Finally, we stress the importance of keeping a clean separation between trust and reputation in order to understand how they can relate and can support the system.

The proposed methodology uses an extension of the problem frames notation in order to accommodate trust and reputation concepts and relationships among these concepts. Intensive context-awareness is an envisioned property of Future Internet systems, and problem frames fit well due to their focus on describing the context around the system-to-be. Also, the context becomes of paramount importance when analysing trust relationships and reputation information, because most of the valuable sources of information for computing trust and reputation will come from this context.

Given that designing complex systems is error-prone, we have also proposed, as part of the methodology, formal OCL checks for model consistency, trust information elicitation and trust reasoning. All this is packaged in a tool that provides support to requirements engineers.

The outcome of our methodology is a set of requirement artifacts. Given that these artifacts are shaped around the problem frames approach, and that this approach encourages the modularization of the system into domains, the artifacts provide a good starting point for sketching the architecture of the system. Trust and reputation models are decomposed in their constituent elements, which provide developers with sufficient information to implement the models and to integrate them into the system in the next stages of the development life cycle.

Acknowledgments

This research was partially supported by the Ministry of Innovation, Science, Research and Technology of the German State of North Rhine-Westphalia and EFRE (Grant No. 300266902 and Grant No. 300267002). The research has also been partially funded by the Spanish Ministry of Economy through the project PERSIST (TIN2013-41739-R) and by Junta de Andalucia through the projects FISICCO (P11-TIC-07223) and PISCIS (P10-TIC-06334). The third author is funded by the Spanish Ministry of Education through the National F.P.U. Program.

References

[1] A. Alebrahim, S. Faßbender, M. Heisel, and R. Meis. Problem-based requirements interaction analysis. In C. Salinesi and I. van de Weerd, editors, *Proceedings of the ReFSQ Conference*, volume 8396 of *LNCS*, pages 200–215. Springer, 2014.

[2] A. Alebrahim and M. Heisel. Supporting quality-driven design decisions by modeling variability. In *Proceedings of the QoSA Conference*, QoSA '12, pages 43–48. ACM, 2012.

[3] K. Beckers, I. Côté, D. Hatebur, S. Faßbender, and M. Heisel. Common Criteria CompliAnt Software Development (CC-CASD). In *Proceedings 28th Symposium on Applied Computing*, pages 937–943. ACM, 2013.

[4] K. Beckers, S. Faßbender, M. Heisel, and R. Meis. A problem-based approach for computer aided privacy threat identification. In B. Preneel and D. Ikonomou, editors, *APF 2012*, volume 8319 of *LNCS*, pages 1–16. Springer, 2014.

[5] K. Beckers, T. Frese, D. Hatebur, and M. Heisel. A Structured and Model-Based Hazard Analysis and Risk Assessment Method for Automotive Systems. In *Proceedings of the International Symposium on Software Reliability Engineering*, pages 238–247. IEEE, 2013.

[6] K. Beckers, D. Hatebur, and M. Heisel. Supporting common criteria security analysis with problem frames. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 5(1):37–63, 2014.

[7] BSI. Protection Profile for the Gateway of a Smart Metering System (Gateway PP). Version 01.01.01(final draft), Bundesamt für Sicherheit in der Informationstechnik (BSI) - Federal Office for Information Security Germany, 2011. `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/SmartMeter/PP-SmartMeter.pdf?__blob=publicationFile`.

[8] I. Côté. *A Systematic Approach to Software Evolution*. Deutscher Wissenschafts-Verlag (DWV) Baden-Baden, 2012.

[9] I. Côté, D. Hatebur, M. Heisel, and H. Schmidt. UML4PF – a tool for problem-oriented requirements analysis. In *Proceedings of the International Conference on Requirements Engineering (RE)*, pages 349–350. IEEE Computer Society, 2011.

[10] S. Faßbender, M. Heisel, and R. Meis. Functional requirements under security pressure. In *Proceedings of the ICSOFT Conference*. INSTICC, SciTePress, 2014.

[11] D. Hatebur. *Pattern and Component-based Development of Dependable Systems*. Deutscher Wissenschafts-Verlag (DWV) Baden-Baden, September 2012.

[12] ISO/IEC. Common Criteria for Information Technology Security Evaluation. ISO/IEC 15408, International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), 2009.

[13] M. Jackson. *Problem Frames. Analyzing and structuring software development problems*. Addison-Wesley, 2001.

[14] Kirtland, Alex and Schiff, Aaron. On A Scale of 1 to 5: Understanding Risk Improves Rating and Reputation Systems. `http://boxesandarrows.com/on-a-scale-of-1-to-5/`, Jun 2008.

[15] F. Massacci, J. Mylopoulos, and N. Zannone. Security requirements engineering: The si* modeling language and the secure tropos methodology. In Z. Ras and L.-S. Tsay, editors, *Advances in Intelligent Information Systems*, volume 265 of *Studies in Computational Intelligence*, pages 147–174. Springer Berlin / Heidelberg, 2010.

[16] F. Moyano, C. Fernandez-Gago, K. Beckers, and M. Heisel. Enhancing problem frames with trust and reputation for analyzing smart grid security requirements. In *Proceedings of the Workshop on Smart Grid Security (SmartGridSec14)*, LNCS 8448, pages 166 – 180. Springer, 2014.

[17] F. Moyano, C. Fernandez-Gago, and J. Lopez. A conceptual framework for trust models. In S. Fischer-Hübner, S. Katsikas, and G. Quirchmayr, editors, *9th International Conference on Trust, Privacy & Security in Digital Business (TrustBus 2012)*, volume 7449 of *LNCS*, pages 93–104, Vienna, 2012. Springer Verlag.

[18] F. Moyano, C. Fernandez-Gago, and J. Lopez. Towards engineering trust-aware future internet systems. In X. Franch and P. Soffer, editors, *3rd International Workshop on Information Systems Security Engineering (WISSE 2013)*, volume 148 of *LNBIP*, pages 490–501, Valencia, Jun 2013 2013. Springer-Verlag, Springer-Verlag.

[19] L. Rasmusson and S. Jansson. Simulated social control for secure internet commerce. In *Proceedings of the 1996 workshop on New security paradigms*, NSPW '96, pages 18–25, New York, NY, USA, 1996. ACM.

[20] R. Roman, J. Zhou, and J. Lopez. On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*, 57:2266–2279, July 2013.

[21] M. G. Uddin and M. Zulkernine. Umltrust: Towards developing trust-aware software. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, pages 831–836, New York, NY, USA, 2008. ACM.

[22] UML Revision Task Force. OMG Object Constraint Language: Reference, February 2010.

[23] A. van Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. John Wiley & Sons, 1st edition, 2009.