Risk Driven Requirements Specification (RiDeRS) of IT-based Homecare Systems *

Mohammad Zarifi Eslami[1], Brahmananda Sapkota[1], Andrea Herrmann[2], Alireza Zarghami[1], Marten van Sinderen[1] and Roel Wieringa[1]

[1] Department of Electrical Engineering Mathematics and Computer Science
University of Twente, The Netherlands
{m.zarifi,b.sapkota,a.zarghami,m.j.vansinderen,r.j.wieringa}@utwente.nl
[2] Independent Researcher, Germany
andreaherrmann3@gmx.de

Abstract. Use of IT in providing homecare services to elderly people is expected to reduce the workload of care-providers. It is also expected that this will increase the quality of services by providing services round-the-clock and will support independent living of the elderly. However, IT-based care systems can also introduce new types of risks such as those related to availability and accountability. This can possibly lead to a decline of using such expensive IT-based homecare systems in practice. In order to prevent this, we propose a method to identify potential risks of using such a system, and to specify additional requirements of the system to mitigate or prevent these risks. We validate the usability and potential utility of our approach by three experiments using a case from the homecare domain. We discuss whether the proposed approach can be generalized for use in the wider class of adaptive critical systems.

1 Introduction

There is an emerging trend in industrialised countries to use IT-based care services including health monitoring, event-based reminder services and alert service to support independent living of elderly (care-receivers) in their home [1, 18]. The aims are (1) to increase productivity of care-givers, which is needed in the face of the aging population in the coming years, and (2) to improve the quality of care. The European Council recognises improvement of patient safety as one of the benefits of using eHealth systems [29]. However, IT-based care services can also introduce new types of risks.

In the literature, risk is defined differently in different domains [30, 14, 7]. In this paper, we use the simple dictionary definition of risk as the "possibility of loss or injury". We do not assume that probability, loss or injury are quantifiable, since whether they are quantifiable strongly dependant on the environment. However, we do assume that the system under consideration has users who can

* This work is part of the IOP GenCom U-Care project(http://ucare.ewi.utwente.nl), sponsored by the Dutch Ministry of Economic Affairs under contract IGC0816.

be hurt, and so there are risks of loss or injury. Users are biological or legal persons who can gain or lose something by the actions of the system, and include patients, their family, nurses, and other health care actors. This means that the concept of risk is user-dependent: what hurts one user may not hurt another one. And what one user would consider an injury may be considered as acceptable pain by another. This means that risk assessments are context-sensitive, and are not transferable from context to context, but must be repeated for every context.

Earlier work, discussed in Section 4, mainly focuses on privacy and security risks of using an IT-system. We argue that an IT-system is not only subject to privacy and security risks, but also *availability*, *safety*, and *accountability* risks [2]. Materialization of these risks could degrade the quality of life of individual users and can even prevent the adoption of such a system in real-life. Assessment of these risks must be performed up-front, as part of requirements engineering, and must continue during the lifetime of the system to discover and mitigate unknown or unperceived risks during early requirements engineering [24]. We propose a Risk Driven Requirements Specification (RiDeRS) approach to elicit requirements of a system by identifying the risks of using such a system as a first step in the requirements engineering process.

Risk-based requirements engineering complements requirements engineering aimed at positive goals. RiDeRS can be used in early requirements engineering, but we see no reason why it could not be used at any point in the system's life cycle to assess risks and to elicit additional requirements.

In RiDeRS, described in Section 2, we consider users' properties besides their goals to identify a list of possible risks and to specify the requirements which could prevent or mitigate these risks. This also helps other stakeholders (e.g., care centers) to perform risk-benefit analysis and to decide whether or not to use the system. What is new in RiDeRS is that it is a goal-oriented method that aims to balance the assumptions about the environment with the requirements on the system. It focusses on safety and it is simple to use in practice, as we will show in our case studies. We apply RiDeRS in a real-life case from the homecare to illustrate its usability and utility, described in Section 3.

2 RiDeRS

Critical systems such as IT-based homecare systems have stringent availability and safety requirements and a failure can threaten human life [26]. To prevent such undesirable system behaviours, we introduce RiDeRS as a risk based approach as a basis for identifying additional requirements on the system to ensure the safety of end-users while offering ease of use. In the remaining of this section, we describe RiDeRS concepts and process.

2.1 RiDeRS Concepts

As shown in Fig. 1, we assume that there is a *system* which works in an *environment* (including the end-users of the system). The system provides *services* which are consumed by *end-users*. The designer of the system makes *assumptions* about the environment of the system, and specifically about the end-users.

These assumptions guided the design and thus underlie the services. Services have associated *goals*, as foreseen by the designer, which can be achieved **if** the assumptions made by the designer are correct. End-users also have goals, which they try to achieve by consuming the services provided by the system. However, if the assumptions made by the designer are incorrect, an end-user may not be able to achieve his goals. Therefore, assumptions imply a *risk*, namely that they do not match the actual properties of end-users, thus preventing end-users from achieving their goals. Such risks may be identified at any point in the lifetime of the system, and suitable *countermeasures* may be proposed to mitigate the risks. Countermeasures are subsequently implemented in the system or in the environment (possibly in both), resulting in a modified system with a changed specification or a modified environment with changed properties.

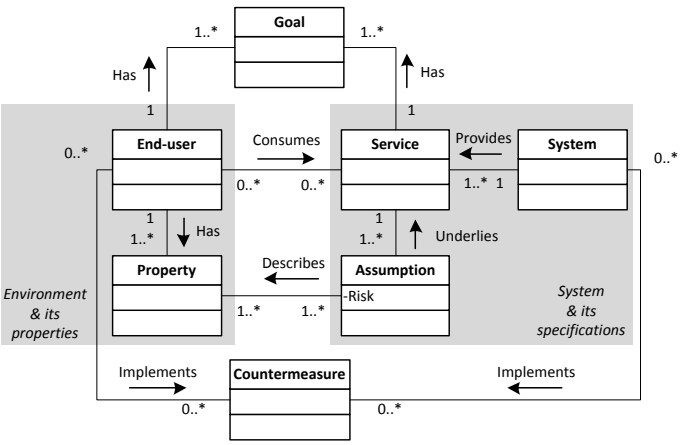

Fig. 1. Conceptual model RiDeRS approach

We classify the assumptions in six categories:

- **Capability assumptions** relate to the capabilities (and limitations) of the end-user. We are only interested in capabilities that are needed to access a service through the provided interface. For example, in the case of a medicine dispenser service, the designer may assume that the end-user can operate a touch panel to control the dispenser. This assumption is not true if the user has Parkinson's disease. The risk is then that the user may not be able to touch the panel in the right positions. Consequently, the system will behave differently than desired.
- **Procedure assumptions** relate to the procedures followed by the end-user to access and consume a service. For instance, in the case of a blood pressure measurement service, the designer may assume that the end-user follows the proper procedure for blood pressure measurement, namely that the measurement instrument is placed around the end-user's left arm and he is in a sitting position. This assumption is not true if the user is an elderly who forgets the correct procedure and attaches the blood pressure cuff to his wrist. The risk is then that the measured values are not precise or reliable.

- **Location assumptions** relate to the location of the user carrying a device with sensors that measures a location parameter for a context-aware service. For example, the designer assumes that the location of the user can be determined with a GPS receiver in the user's phone. This assumption is not true if the user does not carry his phone. The risk is then that a location-based service is applied to the wrong location, and the user is not able to consume the service or consumes a service that is not optimal for his needs/goal.
- **Identity assumptions** relate to the identity of the user who uses a device with sensors that measures the necessary user context parameters. In the blood pressure example, the designer assumes that the measured values from a specific blood pressure meter are in fact those for the intended care-receiver. This assumption is not true if the user takes an instrument which does not belong to him. The risk is then that the reported blood pressure values are not correct and could result in incorrect health related decisions.
- **Needs assumptions** relate to the user having needs matching the user context in which the context-aware service is being offered. For example, the designer assumes that the user needs an audio reminder to take a medicine every five minutes until the medicine is taken. This assumption is not true if the user has not forgotten to take the medicine, but he has an urgent task to finish before he can go to the medicine dispenser. The risk is then that the user will turn off the reminder service because it irritates him, and subsequently forgets to take his medicine.
- **Timing assumptions** relate to the availability of the user (at a specific place/time). For any user-triggered service, the user should be available to initiate the service and to consume the offered service. For example, the designer assumes that the user can react to a reminder in say 5 minutes to measure his blood pressure and after 3 reminder repetitions the system raises an alert assuming that the user ignored the reminders. This assumption is not true if the user received the first reminder correctly and acted accordingly, however, because he is far from the place of the measurement, it takes him longer than expected to act. In this case, the system raises an unnecessary alert that irritates the care-givers.

2.2 RiDeRS Process

We define RiDeRS based on the existing quality requirements approach MOQARE [12]. In MOQARE, quality requirements of a system are specified based on identification of misuse cases. In RiDeRS, we also specify quality requirements by identifying risks; however, we specialize as well as extend MOQARE. We consider misuse cases which are caused due to wrong assumptions by the designer about the users (and the use) of the system. Fig. 2 illustrates the RiDeRS process steps in BPMN notation. The process starts by identifying the users of the system and, for each user, the services which a user wants to consume. Then it iterates over identifying, for each user, the assumptions underlying the services, the risks caused by these assumptions, the countermeasures for mitigating these risks, and the changes to the system specifications due to these countermeasures.

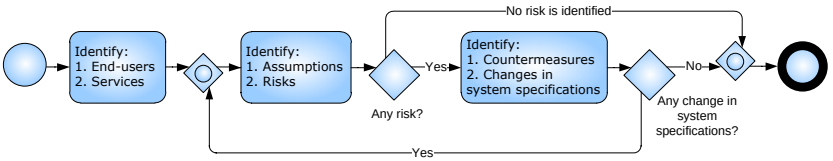

Fig. 2. RiDeRS process

To help the requirements engineer, we formulated a list of questions that the requirements engineer should ask to check whether the assumptions are in fact true with respect to specific users of the system:
- What is the minimum capability that users must have to use this service?
- What procedure must users follow to use this service?
- What does the service assume about the physical location of the user?
- What does the service assume about the identity of the user?
- What assumptions does the service make about the real needs of the user?
- What does the service assume about the time of service consumption?

Regarding each assumption, further questions can be raised. For example: When would these assumptions be violated? Can the system discover they are violated? Can the system still be used when they are violated? Is the system still useful to use if they are violated? What if they are violated? Who is accountable for those hazards that materialize?

After specifying proper countermeasure for each risk, if the countermeasure change system specifications, then the system needs to be analyzed again for new sources of risks. In this regards, we suggest the following "good practices" that we have adopted in our case study: **(a)** For each risk, identify a countermeasure which is implemented by the system in addition to a countermeasure which is implemented by the users. This promotes the consideration of trade-offs between countermeasures to be implemented by the system and by the users, and **(b)** After the identification of countermeasures, reduce the number of different countermeasures by considering their overlap and factor out common aspects.

3 Initial Validations of the Method

To test RiDeRS, we applied it to a real-world case three times in increasingly realistic ways. In this section we describe the case study and the applications.

3.1 Case Organization

The case organization is a care-institution in the Netherlands. This institution consists of residential blocks where elderly can live and receive care services provided through professional care-givers. The aim of this institution is to provide round-the-clock services to their care-receivers and at the same time to enable them to live an independent life as much and as long as possible. As part of the U-Care project, we developed a prototype of an IT-based homecare services platform [32], to be evaluated in this care-institution. This platform has three main components, a *tailoring platform*, a *provisioning platform* (to integrate and

orchestrate the services) and a collection of *application services* such as blood pressure monitoring and medication dispensing. Tailorability of the platform means that some end-users (e.g., care-givers) can change the behaviour of the system without help from technical personnel. This introduces additional risks to the use of the platform, that must be identified from the planned field test. We call the platform plus third-party providers together in short homecare system. We performed a field test of the system in which eight care-receivers and four care-givers used the system. The next three subsections describe three iterations through a risk assessment that was part of the RE done for this field test.

3.2 From MOQARE to RiDeRS

We started our RE by applying the MOQARE method defined by Herrmann and Paech [12] to our case, *without* interacting with domain experts, in order to test whether this method was usable in this case. We slightly adapted MOQARE to our case by starting from end-user goals rather than business goals, and by focussing on risks rather than misuse cases. In the resulting adaptation of MOQARE, we start with the identification of end-users' goals and constrains, and then we identify risks and countermeasures. We learned that MOQARE could **not** be applied as is and required two changes. First, MOQARE represents user goals, misuse cases, and countermeasures in a tree. This tree became too large to be usable, and we replaced it by a one-dimensional tabular representation in which the first column contained user goals and the second one contained user constraints. Second, after analyzing the results, we could see that the countermeasures themselves could be the source of new risks and we should iterate the process of risk identification and countermeasure generation. So we included this loop in the process. The resulting method was the first version of RiDeRS.

3.3 Lab Test of RiDeRS

We next validated the usability of RiDeRS with eleven IT experts who acted as if they were domain experts. We acted as consultants and first explained the subjects the objective of applying RiDeRS. Then we explained how to apply RiDeRS by showing them an example where we apply RiDeRS to *care-givers* as a user and presented them with the corresponding table. Next, we provided them a table to fill the possible risks and countermeasures considering *care-receivers* as a user. The table was pre-filled with the labels of goals and constraints of care-receivers. The task of the participants was to identify possible risks and their countermeasures. In addition to risks and countermeasures, the participants were asked to add more constraints to the table which they deem plausible.

The experiment lasted approximately 30 minutes and all participants succeed in filling in the tables with possible risks. Most of the countermeasures identified by IT experts were ones that should be provided by the system. For instance, we gave "lack of IT knowledge and lack of interest in interacting with the system" as an example of a constraint imposed by care-receivers. Based on this constraint, an IT expert then identified a risk of "not being able or not willing to confirm

to the system of receiving services whenever it is needed". He provided a countermeasure, which is: "the system should support some other ways of checking (e.g., video observation) for receiving the services by care-receivers".

We took away two lessons from this experiment. First, the IT experts conceptualized the risk analysis in terms of capabilities and limitations of end users, and we replaced the concept of a constraint imposed by users, with that of (possibly limited) capabilities of users. Second, our tabular representation of risks and countermeasures was still not usable and we replaced it with a two-dimensional table format. The table lists one or more system services vertically, and for each service lists the assumptions that the environment should satisfy for the system service to be consumed by an end user. Horizontally, the table lists properties some users actually have. In the table entries, we list whether this property is a risk w.r.t. the assumption and if so, what the possible countermeasures could mitigate or eliminate this risk.

Finally, because the method now focussed on user capabilities, we restructured the method to follow the assumption-based reasoning from KAOS [31] and Jackson's problem-frame approach [15]: If the environment satisfies assumption A and the system satisfies requirement R, then the system will contribute to goal G. Our assumptions are assumptions about (limited) end-users capabilities, the requirements are countermeasures to mitigate risks associated with these limited capabilities, and our goals are end-user goals to perform some health-related tasks. This updated method was used in the field test of RiDeRS, described next.

3.4 Field Test of RiDeRS

We validated the usability and usefulness of this updated version of RiDeRS in the case study with care-givers as domain experts. As with the earlier IT-experts, we explained to the care-givers the application of RiDeRS and how to apply it by showing them the *care-giver* example. We asked them to fill in possible risks and countermeasures considering the *care-receiver* as a user. Four care-givers participated in the experiment for an hour.

We made two observations based on this field test. First, it turned out that care-givers could not imagine any risks when the goals and user capabilities were stated abstractly. For example, as a goal, we initially specified "the care-receiver use vital sign monitoring services" and as a capability "he has some disabilities". We needed to make this concrete and specific, as in for example "the care-receiver uses a medication dispenser service which requires pressing a button" and the capability "he has vision impairment", so that they could identify risks.

The second observation is that in comparison with IT-experts, the care-givers usually provided user-based countermeasures rather than system-based countermeasures. For example, for the capability "lack of IT knowledge and lack of interest in interacting with the system", a care-giver identified the risk of "fear of using the system", she identified as countermeasure "training the care-receivers in using a tablet PC and encouraging them to use it by providing some card game applications to play on the tablet PC". Thus, working with domain experts in

the field yielded not only countermeasures that were new system requirements, but also countermeasures that were requirements that end-users must satisfy.

The field test provided yet another lesson, namely that there are more assumptions relevant for a risk assessment than just the capabilities and limitations of end-users. Analyzing our data, we identified the assumptions listed in Section 2.1. We should say at the outset that no risk assessment can be complete, and we do not claim completeness for this one either. However, we do claim that using RiDeRS made it possible to identify more risks than would otherwise have been possible, although empirical comparison with other risk assessment methods remains to be done.

4 Related Work

In this section, we compare the meta models of other methods with RiDeRS. We distinguish the following three kinds of methods:

1. *Vulnerability-oriented methods* - Some methods, such as MOQARE [12], consider the system environment only in terms of "vulnerabilities". Vulnerabilities are properties of the system, users or other actors in the environment which can cause risks or increase their probability or extent of damage. Calling them all "vulnerabilities" means treating them jointly, as one group of concepts, although they are not. When treating vulnerabilities as one category, the question about vulnerabilities in the elicitation process plays the role of a very vague trigger, while in RiDeRS the sub-categories offer more guidance for identifying specific vulnerabilities. Some other methods that model vulnerabilities are: van Lamsweerde et al.'s [31], Lin et al.'s [20] problem frames variant, Firesmith's templates [10], the method of the Object Management Group [23], the vulnerability-centric requirements engineering framework of Elahi et al. [9], and RiskREP [11].
2. *Methods which allow modeling vulnerabilities* - The following methods contain concepts which are equivalent to vulnerabilities, although they call them otherwise. ATAM (Architecture Trade-off Analysis Method) [17] identifies Sensitivity Points in an architecture which are critical for the system's quality. The Misuse Case templates of Sindre and Opdahl [25] contain three fields which can contain vulnerabilities, i.e., assumptions, preconditions and related business rules. The attack patterns of Moore, Ellison and Linger [22] contain preconditions for the misuse cases. UMLsec models constraints [16]. FMEA (Failure Mode and Effect Analysis) [27] uses tabular templates to analyze the potential failures in a system or process, each failure's causes and effects. The threat modeling extension of the NFR Framework [8] includes both vulnerabilities and access points. The BSI Standards [3] model complete processes of security management and guide the search for vulnerabilities.
3. *User-oriented methods* - Some other methods do not consider the environment's role but rather focus on the direct interaction between users and system. These consequently mainly identify risks caused by the users. Such methods are: Sutcliffe and Minocha's scenario templates [28], the abuse case model of McDermott and Fox [21], the NFR Framework of Chung et al. [5], Tropos [4], the EMPRESS Quality Models [6], and SecReq [13].

5 Conclusions

This paper presented the conceptual framework RiDeRS and a method for systematic elicitation of documentation of risk-driven system requirements in a tabular form. The RiDeRS approach is based on elicitation of risks that arise from a mismatch of assumptions made by the designer of the system concerning the environment in which the system will be used. Application of RiDeRS leads to the identification of countermeasures which can prevent or mitigate these risks. For the domain of homecare, we focus on risks arising from the interaction of users with the system. RiDeRS results in a table which presents a user's *used services* to achieve *goals*, the services' *assumptions*, *risks* arising due to those incorrect assumptions, and *countermeasures* to eliminate or mitigate these risks.

RiDeRS was demonstrated by applying it to a homecare system in an illustrative case study. IT-based homecare systems are critical systems in which malfunctions or incorrect usage can threaten the health and/or life of elderly persons. Moreover, users of the homecare domain, such as care-givers and care-receivers, have specific capabilities and limitations. If these do not match with the assumptions under which the homecare system was designed, this can be a source of risks, and therefore a risk-based requirement engineering method, such as that proposed by RiDeRS, must be utilized. It is important to note that such a method should explicitly and systematically elicit risks and countermeasures.

As a future work, we will observe the system's behavior in the field test and seek to identify the actual risks. Then we can do reverse engineering, by writing those risks in a RiDeRS table, try to identify their associated service assumptions and user properties. We should also try to understand why we had not identified these risks. Based on these results, we can further identify where and why the RiDeRS approach requires further refinement. Other future work includes an experimental comparison of the risks identified by RiDeRS and the risks identified by other methods such as Leveson's STAMP method [19].

References

1. Amigo: Ambient intelligence for the networked home environment project (2008), available at: http://www.hitechprojects.com/euprojects/amigo
2. Avizienis, A., et al.: Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Trans. Dependable Secur. Comput. 1(1), 11–33 (2004)
3. Bundesamt für Sicherheit in der Informationstechnik: BSI-Standard 100-3: Risikoanalyse auf der Basis von IT-Grundschutz (2008)
4. Castro, J., Kolp, M., Mylopoulos, J.: Towards requirements-driven information systems engineering: the Tropos project. Inf. Syst. 27, 365–389 (2002)
5. Chung, L., Supakkul, S.: Capturing and reusing functional & non-functional requirements knowledge: A goal-object pattern approach. In: IRI. pp. 539–544 (2006)
6. Dörr, J., et al.: Quality Models for Non-functional Requirements. IESE-Report Nr. 010.04/E (2004)
7. Duffus, J., Brown, S., Fernicola, N.: Glossary for chemists of terms used in toxicology. International Union of Pure and Applied Chemistry 65, 2003–2122 (1993)
8. Ebenezer, A., et al.: Security Threat Modeling and Analysis: A Goal-Oriented Approach. In: 10th Intl. Conf. on Software Engineering and Applications (2006)

9. Elahi, G., Yu, E., Zannone, N.: A vulnerability-centric requirements engineering framework: analyzing security attacks, countermeasures, and requirements based on vulnerabilities. Requirement Engineering 15(1), 41–62
10. Firesmith, D.G.: Analyzing and Specifying Reusable Security Requirements. In: Solid Freeform Fabrication Sym. pp. 507–514 (2003)
11. Herrmann, A., Morali, A.: Riskrep: Risk-based security requirements elicitation and prioritization (extended version) (2010), http://doc.utwente.nl/72721/
12. Herrmann, A., Paech, B.: MOQARE: Misuse-oriented Quality Requirements Engineering. Requirements Engineering 13, 73–86 (2008)
13. Houmb, S.H., et al.: Eliciting security requirements and tracing them to design: an integration of Common Criteria, heuristics, and UMLsec. Requirements Engineering - Special Issue on: Security Requirements Engineering 15, 63–93 (2010)
14. ISO/IEC: Information technology, Security techniques, Guidelines for the management of IT Security: Concepts and models. Intl. Std. 13335-1 (2004)
15. Jackson, M.: Problem Frames: Analysing and Structuring Software Development Problems. Addison-Wesley (2000)
16. Jürjens, J.: Secure systems development with UML. Springer (2005)
17. Kazman, R., Klein, M., Clements, P.: ATAM: Method for Architecture Evaluation. CMU/SEI-2000-TR-004, Software Eng. Inst., Carnegie Mellon University (2000)
18. Korhonen, I., Parkka, J., Van Gils, M.: Health monitoring in the home of the future. Engineering in Medicine and Biology Magazine, IEEE 22(3), 66 – 73 (2003)
19. Leveson, N.: Engineering a safer world: Systems thinking applied to safety. MIT Press (2012)
20. Lin, L., et al.: Analysing Security Threats and Vulnerabilities Using Abuse Frames. Technical Report No. 10, The Open University, UK (2003)
21. McDermott, J., Fox, C.: Using Abuse Case Models for Security Requirements Analysis. In: 15th Annual Computer Security Applications Conference. pp. 55–65 (1999)
22. Moore, A.P., Ellison, R.J., Linger, R.C.: Attack modeling for information security and survivability (2001)
23. Object Management Group: UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms (2004)
24. Rakitin, S.R.: Coping with defective software in medical devices. Computer 39, 40–45 (2006)
25. Sindre, G., Opdahl, A.L.: Eliciting security requirements with misuse cases. Requirements Engineering 10, 34–44 (2005)
26. Sommerville, I., Sawyer, P.: Requirements Engineering: A Good Practice Guide. John Wiley & Sons, Inc., 1st edn. (1997)
27. Stamatis, D.H.: Failure Mode and Effect Analysis - FMEA from Theory to Execution. American Society for Quality Press (2003)
28. Sutcliffe, A.G., Minocha, S.: Scenario-based Analysis of Non-Functional Requirements. In: 4th Intl. Workshop on REFSQ. pp. 219–234 (1998)
29. The Council of The European Union: Council conclusions on a safe and efficient healthcare through ehealth. In: Official Journal of the European Union (2009)
30. UN-ISDR: Terminology on Disaster Risk Reduction. Geneva (2009)
31. van Lamsweerde, A., et al.: From System Goals to Intruder Anti-Goals: Attack Generation and Resolution for Security Requirements Engineering. In: Workshop on Requirements for High Assurance Systems. pp. 49–56 (2003)
32. Zarifi Eslami, M., et al.: Flexible homecare application personalization and integration using pattern-based service tailoring: Supporting independent living of elderly with it. In: 11th Intl. Conf. on CIT. pp. 467 – 474 (2011)