Requirements engineering methods for an Internet of Things application: fall-detection for ambient assisted living

Sofia Meacham[1], Keith Phalp[1]

[1]Faculty of Science and Technology, Bournemouth University,
Fern Barrow, Poole, Dorset, BH12 5BB, UK
smeacham@bournemouth.ac.uk, kphalp@bournemouth.ac.uk,

Abstract

In this paper, hybrid requirements modelling approach is proposed for the Internet of Things (IoT) application of fall-detection for ambient assisted living.

In describing our approach an example case study that covers development phases from the requirements identification and modelling to the design and implementation of such systems is provided. The combination of different formalisms in order to achieve our goal, notably Volere templates for requirements documentation, Use Case diagrams and SysML diagrams for specification, SysML Block Definition Diagrams for system design and implementation was investigated. The suitability and advantages/disadvantages of each modelling approach as well as their combination were analysed and presented. Last but not least, future research with more applications and further design aspects are presented.

Keywords: requirements, SysML, IoT, fall-detection

1.0 Introduction

In this paper, a systematic approach for the most important parts of projects that deal with the design and development of Internet of Things systems is proposed. The case study used was a fall-detection system for ambient assisted living, which can be categorized as part of the general Internet of Things applications.

According to the World Health Organization [1] approximately 28-35% of people aged 65 and over fall each year increasing to 32-42% for those over 70 years of age. The situation is getting worse due to the fact that elderly people often have to stay alone for long periods of time either in their own home environments or in care homes. In this context, automatic fall-detection systems can enable triggering of an alert (manual or automatic) in an emergency situation, thus enabling help when it is required, reducing deaths from falls and consequently increasing the personal feeling of security of elderly people. There are several available fall-detection systems, each of which address some of the requirements, both for indoor [2] [3] and outdoor environments [4]. However, the requirements for these systems are rarely properly defined and formulated.

In this paper, an attempt to properly define the requirements for these types of systems through the use of a variety of different formalisms will be initiated.

First, Volere templates have been chosen for the initial steps. Among all the steps of the Volere requirements engineering process, the use of Volere templates is more well-known and widely-adopted. The structured view of the requirements document allows for a systematic and methodical way of defining the requirements using a document format.

Moving from a document format to a diagrammatic modelling representation, Use Cases have been used for software requirements for many years. There are several advocates that support them as well as literature that criticizes them, and indeed, the authors have been part of a drive to improve their utility. However, despite such arguments, standard use cases remain the most commonly used modelling technique for requirements engineering over the years, arguably due to their simplicity and understand-ability [5]. On the other hand, SysML has emerged as a new standard for system design and is replacing the traditional UML based approach. It is an extension of UML that moves the designer from the software focus of UML towards a more systems focussed approach and is expected to have widespread use among system engineers.

In this paper, a hybrid requirements modelling approach that utilises and combines the advantages offered by each formalism (Volere templates, Use Cases, SysML Requirements diagrams) is proposed in order to increase efficiency and quality of the design and implementation results.

The proposed approach consists of the following steps: The first step is to start from Volere templates to present the requirements in an organised English document; the second step consists of creating Use Cases; the third step representing the structure of the requirements using SysML Requirements diagrams; the fourth step designing the system block diagrams in SysML Block Definition Diagrams; and the fifth step is the system implementation.

To illustrate the approach, in Section 2, a case study will be described. In Section 3, a short description of the model-based design approach and the UML/SysML modelling languages for model-based will be initiated. In the following Section, 4, the use of Volere templates is presented and the High-level/ Low-level Use Case and Requirements diagrams are presented and compared. The system design and implementation will be presented in Section 5, whereas the resulting requirements "flow" from modelling to design/implementation will be extracted in Section 6. In Section 7, reflections and evaluation of our approach will be presented. Finally, Section 8 offers some conclusions and suggestions for future research directions.

2.0 Case Study Overview

This case study was set in collaboration between Bournemouth University and the Technological Educational Institute of Western Greece (TWG), the Embedded System Design and Application Laboratory (ESDA lab) of the Computer Engineering and Informatics Department. Specifically, the TWG/ESDA Lab has many years expertise in IoT applications and assisted living systems. The example of their Ambient Assisted Living laboratory (AAL) in Patras provided significant input to this work.

Specifically, in this case study, the requirements gathering and analysis, system design and implementation for an indoor/outdoor fall-detection ambient assisted living system for a UK care home are being considered.

The system will automatically detect various elements of fall-detection such as the elderly person moving out of the care home or the garden area, detection of no movement for an extended period of time (8 hours) or a sudden acceleration such as a fall.

General medical information needs to be kept using electronic means so that patients, their relatives and medical professionals (doctors, nurses, and carers) can access/update and consistently maintain the data. Security and privacy issues should be maintained for medical data and each user of the system should have different privileges in using the stored information. The data should be handled according to, in our case, the relevant UK data protection act and ethics rules and considerations for medical information.

The medical professionals (nurses, carers) should have access to the system for observing the level and recharging the batteries of the fall detection system.

Also, the elderly person should be able to have a choice of communicating directly with corresponding carers, nurses, relatives in addition to the automatic alarms.
In the case of a false automatic alarm, the elderly person should be able to designate that the alarm was false through an appropriately designed interface.

Last but not least, a major requirement for the future would be to add to the system "intelligent" behaviour wherever appropriate. For example, in case a fall is detected, monitoring mechanisms should be increased in order to obtain more information about the criticality of the incident and the patient's medical condition.

3.0 Model-based Design: UML/SysML modelling

Throughout this project, Model-Based Systems Engineering (MBSE) was applied as an approach to the design and development of a number of systems. In MBSE, models take a central role, not only for analysis of these systems but also for their construction. According to INCOSE, the adoption of MBSE has several advantages [6] such as: improved communication among stakeholders, team members through diagrammatic model representations; improved quality through early identification of problems and fewer errors at the integration stage; increased productivity through reusability of existing models and reduced risk through improved estimates and on-going requirements validation and verification. Overall, it has been said to increase productivity and efficiency in the design and development mainly of complex systems.

Specifically in this project, a combination of SysML and UML modelling languages for MBSE within an Eclipse/Papyrus environment was used [7]. SysML was adopted due mainly to its suitability for modelling a wide variety of systems [8]. It was the result of a UML RFP recommendation for system engineers and has been adopted by OMG since 2006. It offers system engineers several noteworthy improvements over UML. SysML reduces the software-centric restrictions of UML and adds more diagram types such as block definition diagrams, internal block diagrams, parametric and requirements diagrams. Owing to the above additions, SysML is able to model a wide range of systems such as hardware, software, information, processes [9]. On the other hand, UML diagrams were still used wherever it was more appropriate for the project.

4.0 Proposed hybrid requirements modelling approach

4.1 Volere templates

For requirements gathering and analysis, defining our requirements was initiated by using Volere templates [10]. Volere templates are part of a widespread requirements engineering process. In our project, they were used in order to provide a systematic way to formulate the requirements documents.

4.2 High-level Diagrams

As a second step, modelling and diagrammatic techniques were used, among which was SysML/UML Use Cases and SysML Requirements diagrams. The choice of combining SysML Requirements Diagrams with the traditional Use Case modelling was deliberate and offered a variety of views to the system requirements. Use Cases provide the actor-based description of the system [11]

whereas Requirements Diagrams gave a diagrammatic view that connected requirements with block diagram implementations, offering a path to traceability/verification as well as providing relationships between requirements.

4.2.1 High-level Use Case

In Fig. 1, we can see a high-level Use Case Diagram that consists of three main categories of Use Cases: Monitor Device which is used to monitor movement, location and battery levels; Manage Alert which creates an alert in case something is wrong; Manage Record which coordinates the storage and maintenance of medical information

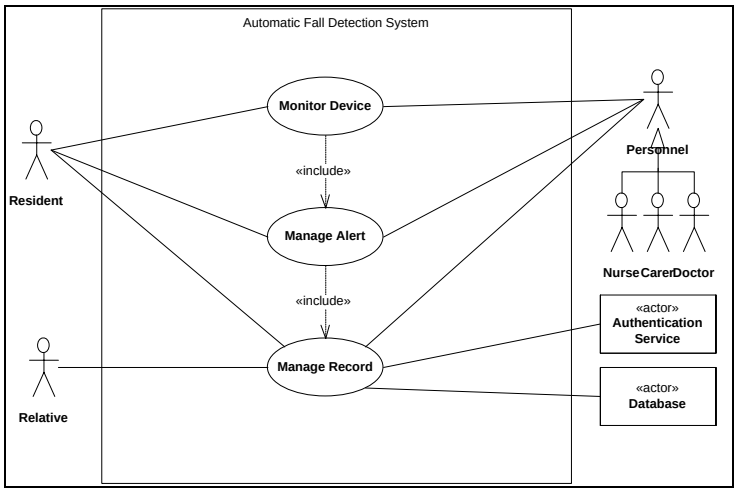

Fig. 1 High-level Use Case Diagram

In this Use Case, the system is presented from the point of view of the main actions that the actors perform (resident, relative and personnel).

4.2.2 High-level Requirements Diagram

In Fig. 2, a high-level Requirements Diagram is depicted that consists of three main categories of requirements: Monitor Environment which is used to monitor movement, location and battery levels; Alert Environment which creates an alert for five cases (manual alert, fall detection, no movement detection, out of range, low battery); Operating Environment which can be indoor, outdoor and 24/7 operating system.

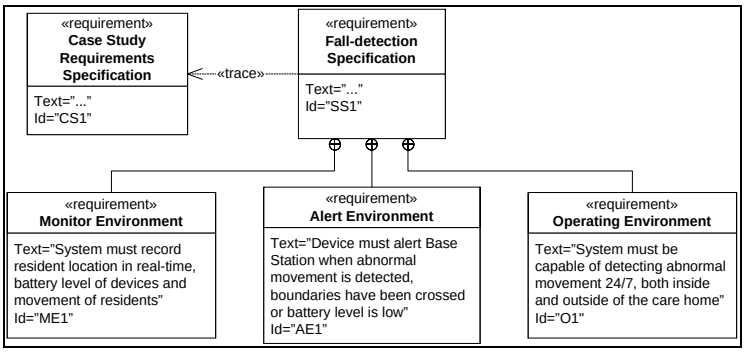

Fig. 2 High-level SysML Requirements Diagram

4.2.3 Comparison of High-level Diagrams

In the above Use Case, the system is presented from the point of view of the main actions that the actors perform (resident, relative and personnel) such as monitor, alert and manage record. On the other hand, in the above Requirements Diagram, the system is depicted from the point of view of the main structural blocks that are required by the system implementation such as monitoring, alert and operating environment.

There is a one-to-one correspondence between the monitor and alert parts of both diagrams. The difference starts when in the Use Case the managing of records is stated explicitly as it is an important action that happens by the actors, whereas in the Requirements Diagram there is a general block which is called operating environment and includes/hides within it the management of records. Last but not least, it is very hard to define which of the two diagrams lies at a higher level of abstraction as both describe the high-level system requirements from a different point of view.

4.3 Low-level Diagrams

4.3.1 Low-level Use Case

In Fig. 3, a low-level Use Case Diagram for Abnormal Condition Detection is presented; specifically Monitor Device. When an abnormal condition occurs, the system detects it using the device attached to the resident. To determine the kind of abnormal condition, the system continuously and transparently evaluates the real-time movement and location of residents. The system runs an algorithm to detect the kind of abnormal condition such as fall, no movement and the location of resident.The system also detects if the condition is a device service alert, for example low battery. The alert is then managed by the Use Case “Manage Alert”.

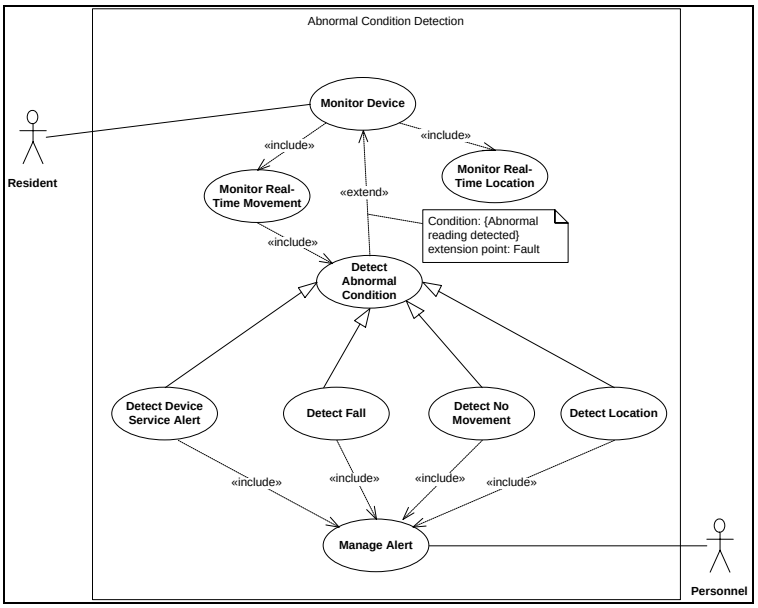

Fig. 3 Low-level Use Case Diagram

4.3.2 Low-level Requirements Diagram

In Fig. 4, a low-level Requirements Diagram is presented that describes the abnormal condition detection requirement. It consists of three main parts: movement, location and communication. The movement and the location are part of the device decision. Note that in this diagram the corresponding non-functional legal requirements are depicted that are part of the system has to follow such as Medical Device Regulation 2012 and Care Quality Commission 2009.

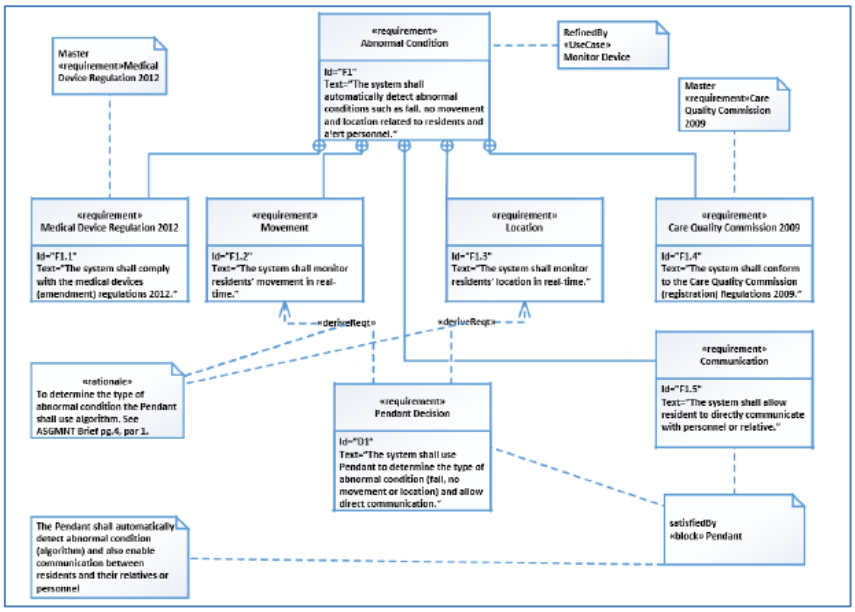

Fig. 4 Low-level SysML Requirements Diagram

4.2.3 Comparison of Low-level Diagrams

In the above Use Case, the system is presented from the point of view of the operations that will have to be performed. Unlike most of the Use Cases, it does not focus on the interactions with actors but rather on the main operations, such as types of abnormal condition (fall, no movement, device service alert). On the other hand, in the above Requirements Diagram, the system is depicted from the point of view of the main structural blocks that are required by the system implementation such as defining movement, location and communication.

There is a one-to-one correspondence between some parts of the two diagrams such as the movement and location blocks. However, the two diagrams are fundamentally different in content. It is very important to note that the requirements diagram includes non-functional requirements such as legal requirements, which are not part of a Use Case diagram. This is an advantage that has been introduced by SysML and is very important for non-functional properties such as legal and security issues. On the other hand, in this case study the Use Case diagram contains all the types of abnormal behaviour as each one of them constitutes a separate operation (use case). This is not explicitly depicted by the SysML Requirements diagram as it is focusing on the structure of the requirements.

5.0 System Design/implementation

In Fig. 5, the corresponding high-level SysML block diagram of the system is presented. A one-to-one correspondence between the requirements in Fig. 2 and the blocks in this diagram is apparent. In addition to this, the "flow" of events is depicted. The Monitor block monitors the system (Operating Environment block) and when something abnormal is detected it raises a Trigger to the Alert block. The Alert block (depending on the specific Trigger that was raised), decides on the required adaptation and sends an Adapt signal to the Operating Environment to perform the changes.

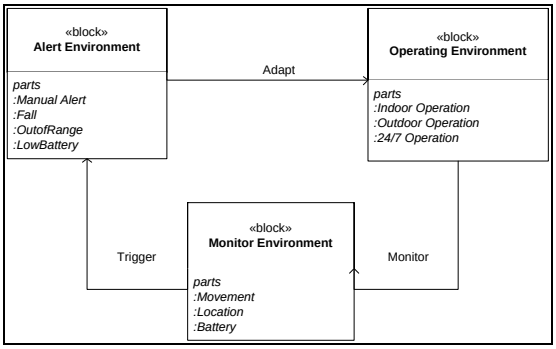

Fig. 5. High-level SysML Block Definition Diagram (BDD)

6.0 Requirements "flow" from modelling to design/implementation

From the above, we contend that the SysML Requirements diagrams most closely match the system design/implementation as they follow the same structural decomposition approach. The correspondence between requirements blocks and blocks of the SysML Block Definition Diagram is very close, whereas the corresponding Use Cases view the system in a more "operations" and actor usage manner.

To recap, the proposed design flow consists from the following steps: The first step is to start from Volere templates to present the requirements in an organised English document; the second step consists of creating Use Cases for the actor, external-use view of the system; the third step consists of representing the structure of the requirements and the non-functional requirements using SysML Requirements diagrams; the fourth step consists of designing the system block diagrams in SysML Block Definition Diagrams; the fifth step is the system implementation.

7.0 Reflections and evaluation of the proposed approach

In this paper, a hybrid approach for requirements modelling was proposed using three different formalisms.

The Volere templates served the purpose of structuring and organising the English document descriptions and are quite a common choice for this purpose.

However, our particular combination of Use Cases and SysML Requirements diagrams represents a novel and effective approach. In [12], Use Case diagrams are defined as a way to "describe the interaction between a system and the environment" whereby Requirements diagrams are described by [13] as "a relationship view of requirements, which address dependencies and allocation of requirements in other software artefacts."

Upon analysis of the requirements for our specific fall-detection case study, a wide range of functional and non-functional requirements such as legal, performance and security issues were identified.

Use Case diagrams were particularly effective at providing a comprehensive summary of functions carried out across the system by elderly patients, carers and doctors. They were also successful at highlighting how the system will behave and adapt in both normal and exceptional cases especially when displaying less common 'out of range' and non-movement usage' scenarios.

Whilst the Use Case diagrams presented a clear, simplistic high-level understanding of functions being performed they have many limitations. Limitations include a lack of technical detail and an inability to show non-functional requirements such as how quickly the alarm would be signalled after the push of the button. This is supported by [14] where Use Cases are specified as 'not being well suited for capturing requirements such as physical, availability and other non-functional requirements'.

In contrast the 'complete' nature of Requirements diagrams meant that showing non-functional requirements such as response times was exceptionally easy. In addition to this requirements diagrams are exceptionally 'traceable' thus making it easy to highlight a hierarchy between technical requirements. For example, when using Requirements diagrams it was possible to illustrate such relationships with the Containment model however these relationships were not possible to depict with Use cases.

In [15], the point that "Requirements should be written at different levels of detail because several stakeholders use them for different purposes" is presented. This

highlights why strengths and weaknesses exist for each modelling technique and is indicative of how they would both be used in combination in the real world.

For instance, Use Case Diagrams are better at depicting functional requirements and user interactions with the system. This is because these diagrams would normally be used during the early stages of the requirements phase to show an understanding between the business analyst and the end users of the system. As the main stakeholder is typically the end user, the level of technical detail is usually kept to a minimum to avoid unnecessary confusion.

In contrast, requirements diagrams are typically created and shared by the business analyst to more technical minded stakeholders such as technical architects and developers. This therefore requires the diagrams to be far more structured and contain more low level detailed technical information.

8.0 Conclusions and future work

This paper has presented a proposed approach to requirements modelling and subsequent design and implementation of Internet of Things (IoT) systems, illustrated by reference to an assisted living case study. Our proposed modelling approach uses a combination of existing tools for modelling requirements, utilising the particular strengths of each, and also providing potential users with notation with which they are likely to be familiar and confident. This solution is of particular importance for the changing and diverse nature of stakeholders, actors and systems involved in an IoT application. Specifically, IoT applications require careful consideration of non-functional requirements that the SysML formalism can provide as well as communication with a diverse number of stakeholders that Use Case diagrams and Volere templates could provide.

Our future plans for research are to apply this approach to more IoT applications as well as to the design and implementation of complex diverse systems such as Systems of Systems (SoS). In addition, requirements traceability and verification that will combine our modelling techniques with formal methods will be investigated. Last but not least, our proposed approach is at its early stages towards developing metrics and methods for ensuring the quality of the design/development for the diverse, ever-changing and adaptive IoT applications of the future.

9.0 References

1. World Health Organization. Ageing and Life Course Unit, 2008. *WHO global report on falls prevention in older age*. World Health Organization
2. Mercuri, M., Garripoli, C., Karsmakers, P., Soh, P.J., Vandenbosch, G.A., Pace, C., Leroux, P. and Schreurs, D., 2016. Healthcare System for Non-invasive Fall Detection in Indoor Environment. In *Applications in Electronics Pervading Industry, Environment and Society* (pp. 145-152). Springer International Publishing.

3. Dong, Q., Yang, Y., Hongjun, W. and Jian-Hua, X., 2015, July. Fall alarm and inactivity detection system design and implementation on Raspberry Pi. In *Advanced Communication Technology (ICACT), 2015 17th International Conference on* (pp. 382-386). IEEE.
4. Busching, F., Post, H., Gietzelt, M. and Wolf, L., 2013, October. Fall detection on the road. In *e-Health Networking, Applications & Services (Healthcom), 2013 IEEE 15th International Conference on* (pp. 439-443). IEEE.
5. Arlow, J. and Neustadt, I., 2005. *UML 2 and the unified process: practical object-oriented analysis and design*. Pearson Education.
6. INCOSE, 2007, *Systems Engineering Vision 2020, v.2.03.* Available at: http://oldsite.incose.org/ProductsPubs/pdf/SEVision2020_20071003_v2_03.pdf. [Accessed 09 February 2016].
7. Papyrus. 2016. *Papyrus*. [ONLINE] Available at: https://eclipse.org/papyrus/. [Accessed 09 February 2016].
8. Bouabana-Tebibel, T., Rubin, S.H. and Bennama, M., 2012, August. Formal modeling with SysML. In *Information Reuse and Integration (IRI), 2012 IEEE 13th International Conference on* (pp. 340-347). IEEE.
9. Apvrille, L. and Roudier, Y., 2015. Designing Safe and Secure Embedded and Cyber-Physical Systems with SysML-Sec. In *Model-Driven Engineering and Software Development* (pp. 293-308). Springer International Publishing.
10. Robertson, S., Robertson, J., 2013. *Mastering the requirements process: Getting requirements right*. 3rd ed. Upper Saddle River, NJ: Addison-Wesley.
11. Kulak, D. and Guiney, E., 2012. *Use cases: requirements in context*. Addison-Wesley.
12. Bertolino, A., Fantechi, A., Gnesi, S., Lami, G. and Maccari, A., 2002, September. Use case description of requirements for product lines. In *Proceedings of the international workshop on requirements engineering for product lines* (pp. 12-19).
13. Ozkaya, I., 2006, September. Representing requirement relationships. In *Requirements Engineering Visualization, 2006. REV'06. First International Workshop on* (pp. 3-3). IEEE.
14. Friedenthal, S., Moore, A. and Steiner, R., 2014. *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann.
15. Soares, M.D.S. and Vrancken, J., 2007, October. Requirements specification and modeling through SysML. In *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on* (pp. 1735-1740). IEEE.