

On the Evolution of Requirements Engineering

Alistair Sutcliffe

School of Informatics, University of Manchester

PO Box 88, Manchester, UK

ags@manchester.ac.uk

1. Introduction

In this paper I argue that the nature of requirements engineering will fundamentally change in the near future as software applications become more intelligent, ubiquitous and mobile. This means that the relationship between the real world and the designed machine will change because the machine will move in the world and know more about the world so it can adapt and react to it. This challenges conventional concepts of requirements as known entities which could be specified and implemented in advance of use.

The changing nature of requirements has been acknowledged for some time, usually in the context of design exploration, where users only “know what they want when they get it”. The co-evolution of software in response to shifts in the environment was the motivator behind Lehman’s models [1]; while the need to adapt changing requirements was proposed by Fickas and Feather [2] for evolving systems to fit with changing worlds, which implied requirements for monitoring and interpreting functions. Fischer’s distinction between adaptable and adaptive systems [3] also implies requirements for configuration facilities and intelligent adaptive processes. In adaptable systems the boundary can be set by the designer and user in different locations to provide a range of configurable solutions. In adaptive systems the machine can change its own behaviour and appearance as a consequence of monitoring the world, e.g. intelligent training systems adapt to the learner by providing different pedagogical strategies, sets of knowledge, interactive tools, etc.

To address the problems of requirements for ubiquitous applications [4] I proposed a method for personal and contextual requirements engineering which captures requirements for processes that enable adaptation to individual users, or location-aware functionality for

mobile and context-aware applications. However, the PCRE method only addressed the process for requirements analysis and did not deal with intelligent applications. To address requirements engineering for adaptable, collaborative and intelligent systems, I argue that a conceptual framework will be necessary to focus on *meta-requirements* which will be necessary for design machines to deal with change and awareness of the world in which they operate.

2. Meta-Requirements Framework

The implications of meta-requirements can be examined by critiquing Jackson’s problem frames [5] framework which describes the dependency relationships between requirements, the real world domain and the designed machine, and has had a considerable influence on RE research. I argue that, while this vision is undoubtedly valuable, it can not account for the diversity of requirements found in adaptable, collaborative and intelligent applications. Software machines are embedded in the domain which they serve. The machine has to detect events in the world and respond to them, leading to either taking action in the world or providing information for people to take action. The dependencies between software machines and the environment can be described in Required Behaviour and Commanded Behaviour problem frames, although no single problem frame maps neatly to the monitoring problem; see figure 1. Monitoring becomes a general problem for any application which is mobile, context-aware or adaptive. It therefore merits a generic model that encapsulates the implied requirements inherent in adaptive and context-aware systems.

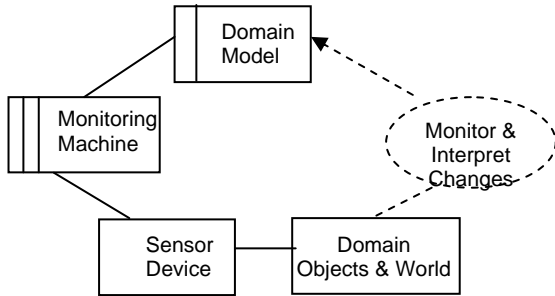


Figure 1. The monitoring problem expressed as a problem frame context diagram with dependencies between the components.

Requirements for context-aware applications [4], can be described in a generic problem model for monitoring as a set of collaborating objects [6] which presents a requirements view that can be mapped to object oriented design in the Observer pattern in the GOF patterns collection [7]. Figure 2 illustrates the object sensing problem model in UML format.

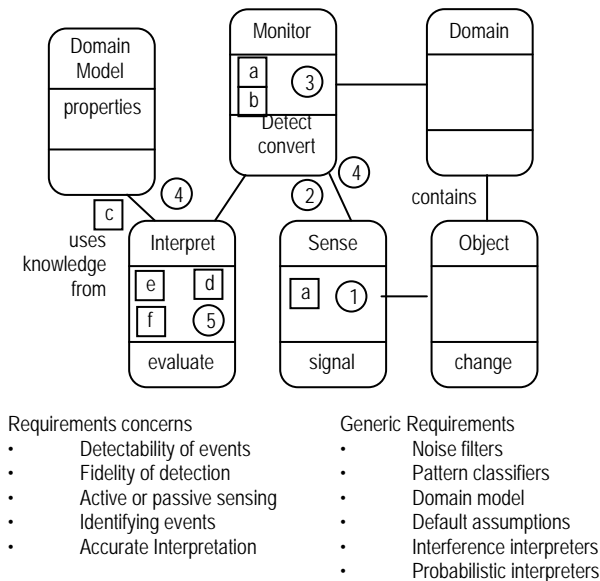


Figure 2. Object sensing model as a set of collaborating objects in UML, with requirements concerns and generic requirements as partial solutions.

The essence of the problem is a sensing device which captures changes in the world, linked to software which converts signals into a digital format for processing and interprets changes into meaningful events. To do so it needs a model of the world, or at least of the expected events. The problem model (see figure 2) is composed of

two classes: one detects signals and maps them to a processable form, and the second interprets the signals as meaningful events. The third component is a model of the world used for interpretation. Interpreter classes can become increasingly sophisticated as the domain model becomes more complex, with complex methods that enable the machine to adapt more intelligently to its environment. However, successful adaptation depends on solving three problems: acquiring and maintaining accurate domain models, correct inference about how to adapt given a particular context and the modus operandi of adaptation. So software machines require models of the world, both for interpreting it and for taking action on the world. Models of the world impose a new class of requirements – which I call *meta-requirements* – since they are not directly part of the functional requirements requested by the users; instead they are requirements which will enable us to build a better and more responsive solution. Meta requirements can be specialised according to the type of context awareness problem, e.g. spatial content, change over time, awareness of the user, groups of users or societies. These meta-requirements need to be elicited and validated early in the process by storyboards and scenarios illustrating a range of adaptive behaviours and contexts. Acquiring and maintaining accurate domain models leads to further meta-requirements which are necessary to specify monitors of domain models, and processes for updating them. This can become a recursive trap of endless monitoring or monitors, although in practice resources enforce limits on the sophistication of awareness a machine (or human) can have about their environment. The theory of mind is one statement of human limitations.

Meta-requirements and more mundane monitoring requirements problems are annotated on the sensing model, to draw attention to concerns such as detectability of changes, active or passive sensing, and data fusion from multiple sources. Problems are mapped to generic requirements which point towards solutions via links to design patterns, algorithm libraries, etc.; this generic problem model not only helps analysis of an application by giving a ready-made abstract template describing the problem, but also points the requirements engineer towards solution knowledge. For example, interpreting changes into meaningful events, solutions depend on the modality of data capture, e.g. in numeric data, data mining algorithms (clustering, association, pattern recognisers, classifiers) are appropriate [8]; for language-based media, text-mining algorithms (domain-specific templates, or parsing-semantic analysis) are available [9]; while for image media, image recognition algorithms and model-based scene interpretation can provide the solution [10].

On the output boundary, the machine also needs a model of the world that it seeks to interact with. This

model may be sophisticated and include detailed topography of a domain and possible behaviours of the effector device. For example, in robotic applications a model is necessary to plan actions and how to move, grip and interact with the physical world. Less directly coupled are control systems applications which interact via physical devices, e.g. motors, switches, and other actuators. Required Behaviour and Commanded Behaviour problem frames describe these dependencies well but they do not address the problem of domain model acquisition and maintenance, except by recourse to workpiece frames as model editors. In mobile and context-aware systems, the model is essential for planning the response as well as for determining the action to take. Models of the user are necessary to customise the machine's output to an individual's needs; furthermore, information itself may need to be output as an interactive model in decision support and simulation tools.

3. Conclusions

The implications of meta-requirements are that we need to focus less on conventional functional requirements but more on questions about what the machine should know about its environment and how it can use the knowledge it possesses to adapt to the world. Other questions concern the range of functional requirements which could be part of the application configuration process. While the latter has been addressed in requirements methods for product lines and ERP configuration methods, requirements analysis for domain and user models held by computer systems has received little or no attention. Ideally the more knowledge possessed by the machine, the more accurate and sophisticated its inferences could be. However, acquisition of domain knowledge incurs a cost and is a well known bottleneck in RE and the development of intelligent systems. One approach to finessing the problem is to employ machine learning, so this provides another twist to the meta-requirements concept: we need to ask whether it is necessary for the machine to learn (volatility and knowingness of the world); or how the machine should learn with respect to the application domain (e.g. classifiers, BBNs, neural nets, explanation-based learning, etc.).

In conclusion, this paper has proposed a definition of meta-requirements, as requirements for maintaining the machine's ability to interpret and act in a domain, in contrast to functional requirements which are responses to the user's goals or logical consequences of interacting with a domain. As software becomes more intelligent, context-aware and adaptable, the type of requirements and the way we capture them will also need to adapt to future worlds which will become harder to anticipate.

References

- [1] M.M. Lehman and J.F. Ramil, "Software Evolution in the Age of Component Based Systems Engineering," *IEE Proceedings: Software*, vol. 147, 249-255, 2000.
- [2] S. Fickas and M.S. Feather, "Requirements Monitoring in Dynamic Environments," *1995 IEEE International Symposium on Requirements Engineering (RE '95)*, 1995, 140-147, Los Alamitos CA: IEEE Computer Society Press.
- [3] G. Fischer, "Shared Knowledge in Cooperative Problem-Solving Systems: Integrating Adaptive and Adaptable Components," in *Adaptive User Interfaces: Principles and Practice*, M. Schneider-Hufschmidt, T. Kuehme and U. Malinowski, Eds. Amsterdam: Elsevier Science Publishers, 1993, 49-68.
- [4] A.G. Sutcliffe, S. Fickas and M. Sohlberg, "Personal and Contextual Requirements Engineering," *13th IEEE International Conference on Requirements Engineering*, 2005, 19-28, Los Alamitos CA: IEEE Computer Society Press.
- [5] M. Jackson, *Problem Frames: Analysing and Structuring Software Development Problems*, Harlow: Pearson Education, 2001.
- [6] A.G. Sutcliffe, *The Domain Theory: Patterns for Knowledge and Software Reuse*, Mahwah NJ: Lawrence Erlbaum Associates, 2002.
- [7] E. Gamma, R. Helm, R. Johnson and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Reading MA: Addison Wesley, 1995.
- [8] O. Maimon and L. Rokach, *Data Mining and Knowledge Discovery Handbook*, Berlin: Springer, 2005.
- [9] S. Ananiadou and J. McNaught (Eds), *Text Mining for Biology and Biomedicine*, Norwood MA: Artech House, 2006.
- [10] L.I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*, New York: Wiley, 2005.