The Challenges of Requirements Engineering in Mobile Telephones Industry

Alessandro Maccari
Nokia Research Center
P. O. Box 407
FIN 00045 – NOKIA GROUP
Mobile: +358 40 749 9074
Fax: +358 9 4376 6308
alessandro.maccari@nokia.com

Abstract

Requirements engineering ranks as one of the most difficult and error-prone phases in the life cycle of devices such as mobile telephones. It is of critical importance because of the highly dynamic market and the constant evolution of product features. If carried out properly, it shortens development time and allows to build products that respond to the market needs. However, it is too often regarded as useless and overly time-consuming. An outlook on the state of practice allows to identify space for improvement of the requirements engineering process (REP). I propose three main challenges that stand on the way towards an optimal requirements engineering practice in our sector. A higher degree of co-operation between the industry and the research world is essential in order to achieve success in this informal yet critical phase of product development.

1. Requirements in mobile telephones

Wireless (mobile) telephony has experienced tremendous evolution in the past few years, and the products, once mere voice communication devices (portable telephones) with minimum extra functionality, have turned into complex wireless computing units.

An example is the Nokia 9100 Communicator [10]: its relevant computing capabilities allow to provide dozens of services including Short Messaging System functions (SMS, only in GSM products), web browsing functionality and wireless imaging.

The user interface has evolved to support graphical displays, while usability has increased as mobile terminals, once expensive tools for very special needs, are used with increasing frequency.

An increasingly complex software architecture has evolved [7] in order to support compatibility between different OS platforms (such as Microsoft's Windows CE and Symbian's EPOC), enable new services such as the Wireless Application Protocol for wireless information services [14], and an ever-growing number of other different services (e.g. intelligent networks).

1.1. The challenges of REP

Mobile terminals are performance critical, software-intensive, embedded systems, and suffer from all drawbacks typical of nomadic computing devices [11]. Special issues arise during software development, particularly regarding architecture [12], development methods [1] and requirements management.

In particular, requirements engineering constitutes the essential starting point for any software system development. Without a well-structured REP, the resulting system is likely to reveal wrong features, not meet market needs, be out of schedule and bug-infested. One of Bob Glass's recent books [2] gets so far as to argue that inadequate requirements management is the most frequent cause of software disasters.

1.2. REP in mobile telephony devices: even more challenging?

Requirements management for mobile telephones is very complex because of the lack of a global standard transmission protocol, the numerous product customisations and the variations due to local standards.

Moreover, several technical limitations affect wireless devices [11], and impose tight requirements that cannot be ignored when designing the products.

A badly structured engineering process can pose severe consequences on several aspects concerning our software development and product success:

- Software architecture: architects, by continuous exchange of information with the system stakeholders, must design and evolve the architecture according to the system-level requirements set. If the requirements process is flawed, the resulting architecture will most likely be wrong or unstable; this could bear extreme consequences on the system integrity and maintainability.
- Complexity: the increasing number of services included in mobile telephone devices, together with the tight constraints on performance (real time devices) and limited resources available (e.g. memory), justify the need of a sound REP.

- Structuring of the features set: the speed of the market and evolution of external aspect, size, functionality, autonomy and price of wireless devices place hard and colliding constraints on the requirements set. A good requirements management enables to define priorities within the requirements set and structure design accordingly.
- Legacy management [5]: all main mobile telephone manufacturers articulate their products into *product lines (families),* or sets of products that share certain common features. Other features vary across and inside the various lines, usually following a somewhat complicated variance pattern. This implies that a relevant part of any new requirements are similar (or equal) to those already tackled in existing products of the same family. New requirements must be adequately related to old ones, in order to avoid building code that performs functionality which has been already implemented. If such a process is carried out effectively, reuse is possible for a large part of the software (see below).
- Reuse: the standardisation of most network-related features (e.g. the signal processing functions) and the relative stability of some hardware elements during evolution (e.g. power devices, user interface) put the base for a wide application of reuse in telecommunications systems [9, 15]. In the mobile telephone market, being capable to effectively reuse software can make the difference with competitors, as it shortens time to market. In practice, however, black box reuse is rare: it is extremely hard to build components for reuse, since it is almost impossible to predict how the requirements will evolve. Most likely, some rework will be needed to reuse components as the requirements change. A clear model of the requirements of each software subsystem may help to properly manage such changes and avoid yet another reuse program failure [Cockburn98, p. 158].

Telecommunication devices could be the most complex systems ever built by man. Too often, the requirements engineering phase is neglected for the sake of speed in launching new products. Improvements of REP in this sector of industry are therefore essential.

2. Issues for REP improvement

The variegation and complexity of requirements engineering makes it hard to state any general principles or guidelines. No analysis of the state of practice that I know of has been carried out specifically for our sector. I propose three main issues connected to software development method, organisation and architecture.

2.1. Requirements engineering is not integrated in the development method

In 1998, our research group published a paper [6] that justified the reasons why a software development method is tightly connected to the aspects of problem domain, structure of product base, software development process and organisation. We have summarised this in Figure 1.

Requirements engineering provides the framework for problem domains. This is why a method that gives weak support for REP is destined to fail.

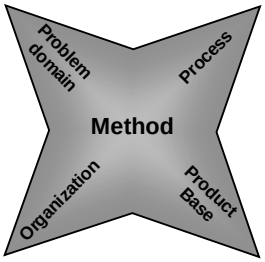

Figure 1. Software development methods and their implications.

At Nokia, some attempts have been made in order to develop a requirements engineering process that takes into account the hierarchy of products in a family. However, the key problem of integrating requirements into a method remains mostly unsolved. The reasons can be connected to the method implications.

- Family requirements are too abstract: it is difficult to express such features in terms of requirements at lower (software components) level because of the complexity of the product architecture *(product domain).*
- Requirements are usually handled by non technical staff (customer service and marketing people), that negotiate the system features (usually, with customers), often unduly neglecting the technical constraints *(organisation).*
- They are difficult to formalise because of the vagueness intrinsic in their expressions. Most refer to user visible features, and are necessarily on a less formal level than, say, hard real-time requirements *(problem domain).*
- While we achieved considerable success in engineering technical requirements, we have so far lagged behind as for managing non-technical

requirements (e.g. usability, performance, reliability). Product family requirements are usually customer visible, and it is not immediate to integrate them into a consolidated development practice *(process).*

It is hard to place all product requirements in a hierarchy, and even harder to predict the evolution of the family structure. These and other factors justify the need of giving requirements management and engineering a well-defined position in software development methods.

It appears that everyone involved in system design and implementation understands that REP is an essential and difficult part of product development. My *first position* is that, notwithstanding the above fact, too little is done to actually integrate requirements engineering into the existing product and software development methods.

Techniques and tools do not lack. A usual first step is the adoption of a use case driven approach [3]. The usage of OOAD methods such as Octopus [1], supported by modelling languages such as the Unified Modeling Language (UML) [13], is also commonly thought to provide a good framework. In this field, unfortunately, the distance between academia and practice sadly reveals to be unacceptably wide.

- The academia and research world regularly produces techniques and notation schemes that are intended to guide us towards formalisation of REP. Too often, there is little proof that
- Almost symmetrically, the industry practice is mostly stuck to ad hoc, informal methods.

Before the usefulness of requirements engineering techniques is scientifically demonstrated, the industry will have no motive to engineer the early parts of software lifecycle. Until then, we should expect little progress.

2.2. Requirements engineering process is not collaborative

Our experience at Nokia teaches us that requirements engineering is often done at the wrong time, by the wrong people, with the wrong techniques and the wrong process.

- Wrong time: formalisation of requirements is needed, but it is usually done after the requirements have been decided and transcribed in an informal and unstructured way (usually, in common language). What is not understood is that formalisation helps not only in designing the product, but also in structuring the requirements set and improving its consistency. While excessive structuring may lead to some wrong decisions and involve excessive effort, it is clear that a greater deal of amount of time and resources should be dedicated to requirements engineering, even after design has started.
- Wrong people: the technical staff have the knowledge necessary to evaluate whether the requirements are achievable and the effort it takes to implement certain features. However, they usually give little contribution in the initial phase of requirements negotiation, which is left to strategy and marketing experts. Moreover, it has happened that some requirements have been added without consulting the customers, with negative effects on the market due to misunderstanding of their needs [proved e.g. in 16]. Customers should always be the main stakeholders of the system, and this is true particularly in the requirements engineering phase.
- Wrong techniques: several requirements elicitation and modelling techniques have been proposed, but they are not applied during requirements elicitation, principally because the people involved in this phase have little technical knowledge (see above: wrong people) and because of the lack of any empirical validity proof for most of them.
- Wrong process: requirements engineering is an essential part of any system's lifecycle, and should be a major part of an incremental, iterative development process (such as the Rational Unified Process [4]). However, the developers cannot usually participate in requirements elicitation but to a limited extent. Thus, they might be forced to design an overwhelmingly complex architecture in an attempt to satisfy requirements that are too stringent. An usual outcome of this scenario is a late reworking of the impossible requirement(s), with overruns in schedule, cost and low product quality. One of the highest resounding examples (which does not belong to the telecom world, but is very representative in its kind) is the Advance Automation System (AAS), that was meant to automate FAA's flight control system and instead resulted into a failure because of an impossible reliability requirement [2, p. 56].

All the above considerations lead to *my second position:* input from the customer (user needs), non technical (such as marketing or strategy) and technical (resources, performance) point of view should all contribute to the set of product requirements. Therefore, representatives of customer, technical and non technical staff should collaborate on the definition of the set of requirements before and during the design phase. Requirements engineering must be a collaborative phase.

2.3. Requirements and software architecture are not connected

With little doubt, the biggest effort in production of mobile telephones goes into software development. In a market where a new product family needs to be launched every few months, software development can involve fatal delays. Quite surprisingly, and contrarily to Bob Glass's findings [2, p.14], our experience tells us that such schedule overruns are more likely to happen because of poor development process and people management (rather than technical) issues.

In most cases, the root of the problem can be found in weak connection between the set of requirements and software architecture. When requirements and architecture evolve separately, the product failure alarm should ring.

Such scenario often realises because the staff developing the architecture has little voice during elicitation of requirements (see 2.2: wrong people). Symmetrically, the people that perform requirements elicitation are not consulted during the architecting phase. Only a few key requirements are considered when developing the architecture, and the rest of the work is left to the component developers.

Hence, my *third position* is that software architecting and requirements engineering cannot be separated. Software architecture is the high level solution to the problems posed by the requirements. The set of requirements should develop hand in hand with the architecture. An architectural team composed by an assorted selection of marketing, customer service and technical staff can help bridge the gap. Alistair Cockburn's proposal in [Cockburn98, p. 129] may be used as a basis for creating the optimal team. This also complies with my previous position, in that it gets people with different functions and background to speak and co-operate around the same table.

3. Evolution

Requirements engineering as an academic discipline is evolving with great speed, and providing diversified solutions to what are commonly believed to be the main industrial problems. However, the industry, especially in the telecommunications sector, is slow and reluctant in adopting and experimenting them.

Little has been published about practice in REP – [8] being one exception. Scarce contribution comes from the telecom industry. This is due to the lack of attention that REP gets, and to the low level of formality of the current processes, which makes the process difficult to describe and the material unsuitable for publications.

The methods that are proposed by researchers should be applied and empirically validated, in order to assess their effectiveness. The academic world should focus on solutions to real, maybe informal, industrial problems.

The industry, on the contrary, should dedicate greater effort to formalisation of the existing requirements engineering process, and focus on closer integration with product development. Higher formalisation is needed, though it should not be abused, since requirements are never completely formal.

Looking at the future, I foresee that an increasing number of projects will experience problems or, in extreme cases, failure due to a bad REP. At Nokia, we are in the process of experimenting a number of methods that provide solutions to the above-mentioned issues. We expect to contribute to the future research and industrial issues in this vital field.

4. Acknowledgements

I thank Jyrki Heikkinen and Juha Kuusela for their comments on requirements and architecture respectively.

5. References

[1]: M. Awad, J. Kuusela, J. Ziegler, *Object-oriented technology for real time systems: a practical approach using OMT and Fusion*, Prentice-Hall, 1996.

[2]: R. L. Glass, *Software runaways*, Prentice-Hall, USA, 1998.

[3]: I. Jacobson, M. Christerson, P. Jonsson, G. Övergaard, *Object-oriented software engineering: a use case driven approach*, Addison-Wesley, 1992.

[4]: I. Jacobson, G. Booch, J. Rumbaugh, *The Unified software development process*, Addison-Wesley, 1999.

[5]: A. Karhinen, M. Sandrini, J. Tuominen, *An approach to manage variance in legacy systems*, in Proc. of the 3rd European Conference on Software Maintenance and Reengineering, Amsterdam, NL, 1999.

[6]: J. Kuusela, A. Karhinen, A. Maccari, *Justification for special purpose object oriented software development methods*, Proc. of the Third International Conference on Object Oriented Methodology WOON98, Saint Petersburg, Russia, 1998.

[7]: J. Kuusela, *Architectural evolution, Nokia Mobile Phones case*, in *Software Architecture*, edited by Patrick Donohoe, Kluwer Academic Publishers, Boston, USA, 1999.

[8]: M. Lubars, C. Potts, C. Richter, *A review of the state of the practice in requirements modelling*, IEEE Symposium on Requirements Engineering, IEEE Computer Society Press, Los Alamitos, CA, USA, 1992.

[9]: A. Maccari, M. Sandrini, *Impact of reuse on software development for telecommunications systems*, Proc. of the European Reuse Workshop 1998, European Software Institute Press, Madrid, Spain, November 1998.

[10]: see http://www.nokia.com/phones/9110/index.html

[11]: E. Pitoura, G. Samaras, *Data management for mobile computing*, Kluwer Academic Publishers, 1998.

[12]: A. Ran, J. Kuusela, *Selected issues in architecture of software intensive products*, Proc. of the Second International Software Architecture Workshop ISAW-2, ACM, 1996.

[13]: see http://www.omg.org/news/pr97/umlprimer.html

[14]: see the WAP forum page at http://www.wapforum.org/

[15]: W. Frakes, [Frakes95]: W. B. Frakes, C. J. Fox, Sixteen questions about software reuse, Communications of the ACM, 6(38), June 1995.

[16]: B. Curtis, H. Krasner, N. Iscoe, A field study of the software design process for large systems, *Communications of the ACM 1988, 31(11).*