Can We Beat the Complexity of Very Large-Scale Requirements Engineering?

Björn Regnell[1,2], Richard Berntsson Svensson[2], and Krzysztof Wnuk[2]

[1] Sony Ericsson, Lund, Sweden
http://www.sonyericsson.com
[2] Lund University, Sweden
bjorn.regnell@cs.lth.se,
http://www.cs.lth.se

Abstract. Competitive development of complex embedded systems such as mobile phones requires management of massive amounts of complex requirements. This paper defines and discusses orders of magnitudes in RE and implications of the highest order of magnitude that we have experienced in industrial settings. Based on experiences from the mobile phone domain we propose research areas that, if addressed successfully, may help beating the complexity of Very Large-Scale Requirements Engineering.

1 Introduction

The complexity and size of software-intensive systems continues to increase, which in turn gives increasingly large and complex sets of requirements. How many requirements can an industrial system development organisation manage with available Requirements Engineering (RE) processes, methodology, techniques and tools? This is hard to know as RE research often falls short in characterizing the scalability of proposed methods. How large and complex sets of requirements do we need to consider when researching new RE technology? We have no complete picture of current industrial practice in terms of complexity of sets of requirements, but we have experiences from industrial cases with enormous complexity where current RE technology have useful but partial effect [4,5,6]. Our objective with this paper is to share some important research opportunities that we have found in our observation of what we call Very Large-Scale Requirements Engineering (VLSRE).

The paper is organized as follows. Section 2 proposes a definition of VLSRE based on the size of a requirement set as a proxy for its complexity. Section 3 provides a case description of the mobile phone domain that illustrates an instance of VLSRE. Section 4 highlights some research opportunities that we through our own industrial experience have found relevant to VLSRE. Section 5 concludes the paper.

2 Orders of Magnitude in Requirements Engineering

Table 1 defines four orders of magnitude in RE based on the size of the set of requirements that are managed by an organisation that develops software-intensive

B. Paech et al. (Eds.): REFSQ 2008, LNCS 5025, pp. 123 – 128, 2008.

Table 1. Three orders of magnitude in Requirements Engineering

Abrev.	*Level*	*Order of magnitude*	*Sample empirical evidence*	*Interdependency management conjectures with current RE technology*
SSRE	Small-Scale Requirements Engineering	~10 requirements		Managing a complete set of interdependencies requires small effort.
MSRE	Medium-Scale Requirements Engineering	~ 100 requirements	[3]	Managing a complete set of interdependencies is feasible but requires large effort.
LSRE	Large-Scale Requirements Engineering	~1000 requirements	[8]	Managing a complete set of interdependencies is practically unfeasible, but feasible among small bundles of requirements.
VLSRE	Very Large-Scale Requirements Engineering	~10000 requirements	[6]	Managing a complete set of interdependencies among small bundles of requirements is unfeasible in practice.

systems. The levels are inspired by the characterisation of orders of magnitude in integration of digital circuits.

We have chosen *numbers of requirements* as a proxy for complexity as we believe that increased numbers of customers, end users, developers, subcontractors, product features, external system interfaces, etc. come along with increased number of requirements generated in the RE process as well as increased complexity of RE. Furthermore, in almost all industrial RE settings that we have encountered, the requirements that are documented are also eventually enumerated and often given a unique identity, allowing a counting of the elements in the set of requirements in a given development organisation. If so, it is fairly easy to give a size figure for a given case that in turn allows for cases to be compared in terms of their order of magnitude (although the average level of detail in the set of requirements needs to be fairly similar for the comparison not to be too speculative).

We suggest based on experience that the complexity of a set of requirement is heavily related to the nature of interdependencies among requirements (see e.g. [2] for an empirical investigation of interdependencies). With a realistic degree of interdependencies among n-tuples of requirements, we hypothesize that the number of interdependencies to elicit, document and validate increases dramatically with increased number of requirements. When shifting from MSRE to LSRE, a typical heuristic for dealing with the complexity of interdependency management is to bundle requirements into partitions and thereby creating a higher level of abstraction where interdependencies among bundles can be managed with reasonable effort. When shifting from LSRE to VLSRE, our conjecture is that even the number of bundles gets too high and the size of bundles becomes too large to allow for interdependency management with desired effectiveness. If the requirements bundles become too large, the interdependency links loose practical usefulness as they relate too coarse grained abstractions.

SSRE and MSRE is a common scale in research papers that seek to validate a proposed method or tool. For example, in [3] the scalability issue is addressed but for a specific tool dealing with only 67 requirements. In this situation it is possible to

enumerate and manage complex relations among requirements even with dense relation patterns. However, we believe that few industrial situations in current system development can avoid stretching beyond SSRE and even MSRE. We have found few examples in RE literature that discusses LSRE (such as [8]), but we believe that LSRE is common industrial practice (confirmed by [1]). We also believe that a significant number of companies that currently face LSRE will grow into the situation of VLSRE as their products grow in complexity, their product portfolio grows in size, and they introduce product line engineering that further drives RE complexity. In the next section we describe one specific case that already has experienced such atransition.

3 A Case of VLSRE

To illustrate the complexity in VLSRE we provide a case description of embedded systems engineering in the mobile phone domain, based on experiences at Sony Ericsson, which has faced a transition from LSRE to VLSRE in the last years, while remaining competitive on the market with a growing number of around 6000 employees. Mobile phones include a wide range of features related to e.g. communication, business applications and entertainment. The technological content is complex and includes advanced system engineering areas such as radio technology, memory technology, software design, communication protocols, security, audio & video, digital rights management, gaming, positioning etc. The complexity of RE is driven by a large and diverse set of stakeholders, both external to the company and internal. Table 2 gives examples of stakeholders that generate requirements.

Table 2. Examples of stakeholders that generate requirements

External Stakeholders	*Internal Stakeholders*
Competitors	Accessories
Consumers of different segments	Customer Services
Content providers	Market research
Legislation authorities	Marketing & customer relations
Operators	Platform development (SW+HW)
Retailers	Product, application & content planning
Service providers	Product development (SW+HW)
Share holders	Product management
Standardization bodies	Sourcing, supply & manufacturing
Subcontractors & component providers	Technology research & development
	Usability engineering

Some stakeholders are counted in billions, such as consumers of different segments, while some are counted in hundreds such as operators. In the case of Sony Ericsson, the requirements that are generated from internal and external stakeholders amount to several tens of thousands, and this is a clear case of VLSRE.

Figure 1 provides a simplified picture of the different types of requirements and their relations. Similar to the case in [3], requirements originating from external stakeholders (called *market requirements*) are separated from but linked to *system requirements* that are input to platform scoping in a product line setting. Market

requirements are mainly generated by operators submitting specifications with thousands of requirements that require statements of compliance. The total volume of market requirements at Sony Ericsson exceeds 10000 as well as the total volume of platform system requirements. In order to make scoping feasible, platform system requirements are bundled into hundreds of *features* that represent the smallest units that can be scoped in or out. In order to support product development the platform capabilities are organised into *configuration packages* that improve over time as more and more features are implemented for each new version of a platform. Products are configured through assemblies of configuration packages according to the rules of how they can be combined based on their interdependencies. All categories of requirements are expressed in natural language text and include a set of attributes according to a requirements data model for a requirements data base implemented in a commercial requirements engineering tool. Based on our experience with the complexity of this VLSRE case we bring forward three key research opportunities in the next section.

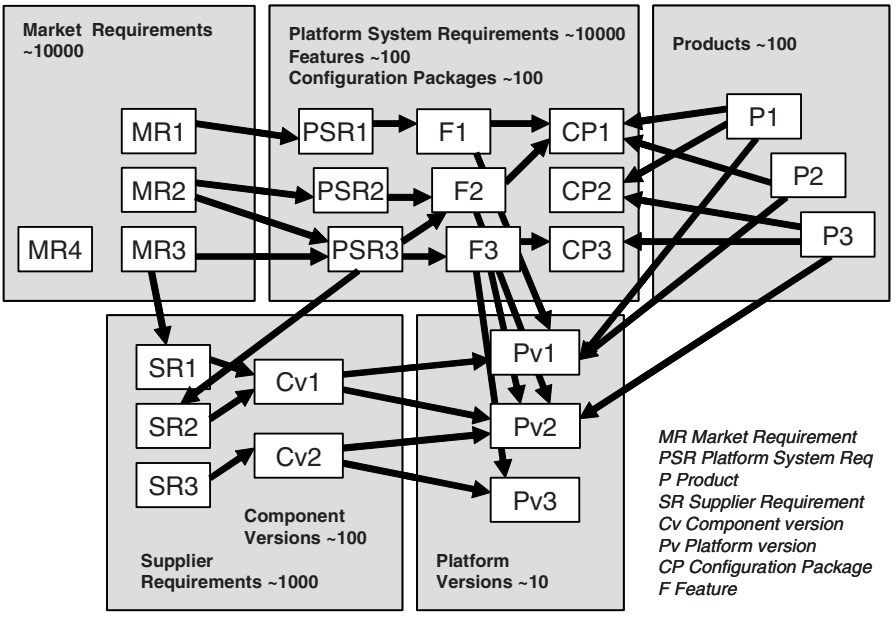

Fig. 1. Orders of magnitude in different artifacts of a specific VLSRE case

4 Three Key Research Opportunities in VLSRE

Based on our experience from working several years in the previously described VLSRE context, we have chosen to highlight three areas where we believe RE research can and should contribute:

- ***Sustainable requirements architectures***: **Fighting information overload.** With the term requirements architecture we mean the underlying structure by which the

requirements are organised including the data model of the requirements with their pre-conceived attributes and relations. In VLSRE, the amount of information that must be managed is immense and not possible to grasp in all its details by a single person. In order to fight information overload we need requirements architectures that are sustainable in the sense that they allow for controlled growth while allowing the requirements engineers in a large organisation to keep track of the myriad of issues that continuously emerge. How should we design sustainable requirements architectures? Which concepts are stable? Which attributes and links are most important to maintain? What is the simplest yet competitive requirements data model?

- ***Effective requirements abstractions***: **Fighting combinatorial explosions.** In VLSRE situations where interdependencies among requirements are critical (such as prioritisation, resource estimation, and change impact analysis) we inevitably stumble on combinatorial explosions, further fuelled by product line engineering that significantly increases the complexity of the requirements architecture. Finding all interdependencies among 20 requirements is possible, but not among 10000. A major vehicle for fighting this is abstraction mechanisms and experience-based heuristics. In interviews with requirements architects at Sony Ericsson we encounter heuristics related to requirements bundling and choice of level of detail, but they still often struggle to find yet another needle in the haystack. Can we empirically characterize the effectiveness of requirements abstractions? How can we empirically investigate human requirements comprehension? How to support humans in navigating and searching massive sets of requirements? How can we make relevant visualisations of different partial viewpoints on immense requirements heaps that hide irrelevant details but highlight important issues for a given decision-making situation? What level of uncertainty and degree of approximation can we tolerate?
- ***Emergent quality predictions***: **Fighting over-scoping.** Given a competitive market and a large and demanding set of stakeholders, there seems to be an inevitable shortage of resources to meet quality expectations. To predict the system level quality aspects that emerge from a myriad of details is very difficult and we have seen a sustained risk of defining a too large scope for platform development partly due to the inherent difficulty in understanding quality requirements and predicting their impact and required development resources. We are beginning to understand how to do roadmapping and cost-benefit analysis of quality requirements in sub-domains [7], but we still struggle with how to manage a holistic view where quality requirements are aggregated to system level. How can we deal with interdependencies among quality requirements? Maybe we can get the scope of functions right, but are the set of functions of adequate quality? How can we with reasonable effort prioritize emergent system qualities when predictions are uncertain?

5 Conclusion

During the last decade we have seen VLSRE emerge as a very demanding challenge. Parts of the embedded systems engineering industry are facing severe problems in coping with the rapidly increasing complexity of the massive amount of information that needs to be managed in order to be competitive on the market. Our conjecture is that we have hit the roof with current tools and we need to mobilise RE researchers to try to beat the

complexity of VLSRE. We should also increase our knowledge of how existing methods, tools and techniques perform in SSRE, MSRE, LSRE and VLSRE respectively, to better understand which methods that are good candidates for use in VLSRE combined with sustainable requirements architectures and effective requirements abstractions. By advancing these techniques and heuristics we might be able to manage the complex task of predicting emergent system quality aspects already in the early stages of the development cycles where opportunities are rising while uncertainties are high.

Acknowledgements. This work is supported by VINNOVA (Swedish Agency for Innovation Systems) within the MARS and UPITER projects. Special thanks to Thomas Olsson for input on numbers and to Even-André Karlsson for input to the visualisation of entity relations in fig. 1.

References

1. Brinkkemper, S.: Requirements Engineering Research the Industry Is and Is Not Waiting For. In: 10th Anniversary Int. Workshop on Requirements Engineering: Foundation for Software Quality. Riga Latvia, pp. 41–54 (2004), URL visited 07/12/2007 http://www.sse.uni-essen.de/refsq/downloads/refsq-10-booklet.pdf
2. Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., Nattoch Dag, J.: An industrial survey of requirements interdependencies in software product release planning. In: Proc. IEEE Int. Conf. on Requirements Engineering, Toronto, Canada, pp. 84–91 (2001)
3. Feather, M.S., Cornford, S.L., Gibbel, M.: Scalable mechanisms for requirements interaction management. In: Proc. 4th Int. Conf. on Requirements Engineering, Los Alamitos, USA, pp. 119–129 (2000)
4. Natt och Dag, J., Gervasi, V., Brinkkemper, S., Regnell, B.: Speeding up Requirements Management in a Product Software Company: Linking Customer Wishes to Product Requirements through Linguistic Engineering. In: Proc. of the 12th Int. Requirements Engineering Conf., Kyoto, Japan, pp. 283–294 (2004)
5. Natt och Dag, J., Gervasi, V., Brinkkemper, S., Regnell, B.: A Linguistic Engineering Approach to Large-Scale Requirements Management. IEEE Software 22, 32–39 (2005)
6. Regnell, B., Olsson, H.O., Mossberg, S.: Assessing requirements compliance scenarios in system platform subcontracting. In: Proc. 7th Int. Conf. on Product Focused Software Process Improvement, Amsterdam, The Netherlands, pp. 362–376 (2006)
7. Regnell, B., Höst, M., Berntsson Svensson, R.: A Quality Performance Model for Cost-Benefit Analysis of Non-Functional Requirements Applied to the Mobile Handset Domain. In: Sawyer, P., Paech, B., Heymans, P. (eds.) REFSQ 2007. LNCS, vol. 4542, pp. 277–291. Springer, Heidelberg (2007)
8. Park, S., Nang, J.: Requirements management in large software system development. In: IEEE Conf. on Systems, Man, and Cybernetics, New York, USA, pp. 2681–2685 (1998)