Overscoping: Reasons and consequences — A case study on decision making in software product management

Conference Paper · October 2010

DOI: 10.1109/IWSPM.2010.5623866 · Source: IEEE Xplore

CITATIONS
13

READS
395

3 authors, including:

Elizabeth Bjarnason
Lund University
54 PUBLICATIONS **507** CITATIONS

SEE PROFILE

Krzysztof Wnuk
Blekinge Institute of Technology
164 PUBLICATIONS **1,127** CITATIONS

SEE PROFILE

Overscoping: Reasons and Consequences – A Case Study on Decision Making in Software Product Management

Elizabeth Bjarnason, Krzysztof Wnuk and Björn Regnell
Department of Computer Science
Lund University
Lund, Sweden
{Elizabeth.Bjarnason, Krzysztof.Wnuk, Bjorn.Regnell}@cs.lth.se

Abstract— **Efficient scope management is a core part of software release management and often a key factor in releasing successful software products to the market. In a case when not all the requirements for the next software product release are known 'a priori' and when new requirements are issued throughout the project, the risk of overscoping by including more functionality than can be implemented increases. In this paper, we report on findings from an empirical interview study about understanding the causes and effects of overscoping in a large-scale industrial set up. Six main causes of overscoping have been identified in this work, complemented by root cause analysis of the causes and concluded by effects of overscoping. The results provide an increased understanding of the scoping activity as a continuous activity and outline risks and issues that can lead to a situation of overscoping.**

Keywords: requirements scoping, empirical study, software release planning, case study

I. INTRODUCTION

Software products and software product management has recently emerged as ways of developing software as a product [9][10][32]. To create successful products, a software product manager [32] has to understand and define stakeholders, elicit and manage requirements, define products portfolio and schedule content and timing of releases of these products [8][31]. In this context, requirements management has been recognized as a key area for software product companies [32], which also brings new challenges to requirements engineering. For example, a software product manager has to cope with a large number of continuously arriving requirements while effectively applying a Market-Driven Requirements Engineering (MDRE) process [17][24]. Moreover, since software products can be more easily changed and updated than other products, the release frequency is higher. This in turn increases the complexity of managing requirements changes and release planning activities. Finally, market needs for "fast rotating" software products tend to change dynamically, affected, for example, by a dynamically changing market situation [11] and evolving technologies. As a result, the selection process of which requirements to include into the next release of the software project, also called *scoping* [29], may change its discrete and manageable nature to a more complex and continuous task of assessing how changes in the domain impact scoping decisions.

An example of a scoping process at a large software company using a Software Product Line approach [21] was visualized and analyzed in our previous work [33] where we have reported that changes in the scope of analyzed projects happened throughout the entire requirements engineering process, constituting both many late scope inclusions and exclusions. As reported in [33], feature exclusions are mainly the results of stakeholders' business decisions or lack of resources. However, our previous work [33] investigates only reasons for removing functionality from an overloaded scope of the project. The interesting question to pose here is what actually causes the situation of setting a scope that is too large, which we call *overscoping* of the project, or in other words "biting off more that you can chew"?

Thus, our objective with this paper is to complement the investigation reported in [33] by identifying reasons for overscoping. To achieve this goal, we have conducted an empirical interview study with nine practitioners working with requirements engineering, software development and product testing at a large company, preceded by a pre-study which indentified possible causes of overscoping, as well as the effects of overscoping at the case company. Our responders confirmed four of five assumed causes of overscoping and pointed out a sixth cause of overscoping, namely unclear vision of overall goal. Additionally, we report root causes of overscoping and provide a list of five possible effects of overscoping mentioned by our responders.

The reminder of this paper is structured as follows: Section II provides related work. Section III provides background information about the context of our industrial case study. Section IV describes the methodology used in this study. Section V describes the results from the interview study. Section VI provides our interpretation of the results and discussion including limitations of this study. Section VII provides conclusions and further work.

II. RELATED WORK

Releasing a successful software product in a market-driven context, where the 'customer' is an anonymous consumer market, is inextricably related to the challenge of meeting those customers' demands [2][14]. Therefore, the release time is also important [6][23][34], or even, for some

cases, more important than the functionality that the newly released product is providing. Since there is no consensus made between a customer and a contractor of the system, the scope of the project has to be set using prioritization techniques based on market predictions and effort estimates [4][16]. Finally, the market is primarily verifying the final products [24].

Requirements in MDRE (Market-Driven Requirements Engineering) are invented rather than negotiated and there is no actual customer that can agree to the requirements [17][22][24]. Moreover, as reported by Lubars et al. it is a challenge to interpret customer requirements [19] and to understand who our customer is, in order to be more responsive to their needs. This challenge is also mentioned by Karlsson et al. [17]. Lubars et al. mention the need for prioritizing requirements and the increased rate of requirements change in MDRE [19]. Karlsson et al. [17] and Regnell and Brinkkemper [24] report managing the constant flow of new requirements and release planning based on uncertain estimates among challenges special for MDRE. Finally, reaching a state of congestion as a consequence of allowing more requirements to enter the MDRE process than can be handled with the available resources has also been mentioned as one of the challenges in empirical studies [17][23].

Release planning is where requirements engineering, or market-driven software product development, meets the market perspective [5]. It is an integral part of software product management, where multiple releases are considered in a release plan [32]. A roadmap document is often used to layout the product releases to come over a time frame of three to five years [24]. The product roadmap is then elaborated by product managers to a set of product requirements for the various releases [24]. This set of product requirements serves as a kernel of the software product to be built by developers. The selection of the 'right' requirements for a certain release is normally preceded by requirements prioritization [16] and cost estimation [15]. The selection process is not a trivial task, due to, for example, most features having dependencies to other features [3], or decisions concerning what to include in the next release of a project are often altered [33]. As a result, the selection process often becomes an uneasy compromise where the development of commitments already made may need to be sacrificed at the expense of wasted effort. Therefore, scope management is considered as one of the core functions of software release planning and a key activity for achieving economic benefits in product line development [29]. Much research has been made in the area of release planning, for example, by proposing release planning methods using linear programming [1], the analytics hierarchy process [28], stakeholders' opinions on requirements importance [16][28] and using linear programming techniques using requirements interdependencies [5].

III. THE CASE COMPANY

Our results are based on empirical data from industrial projects at a large company that is using a product line approach [21]. The company has more than 5000 employees and develops embedded systems for a global market. There are several consecutive releases of a platform (a common code base of the product line) where each of them is the basis for one or more products that reuse the platform's functionality and qualities. A major platform release has a lead time of approximately two years from start to launch, and is focused on functionality growth and quality enhancements for a product portfolio. Minor platform releases are usually focused on the platform's adaptations to the different products that will be launched with different platform releases.

A number of different organizational units within the company are involved in the development. For this case, the relevant units are the *Requirements Unit* that is responsible for planning the scope and managing the requirements, the *Software Unit* that develops the software for the platform and the *Product Unit* that develops products based on the platform releases. Within each unit there are several groups of specialists for different technical areas. They are responsible for the work in various stages of the development process in the case company. For this case, the most essential groups are called *Requirements Teams (RTs) (part of the Requirements Unit)* that elicit and specify system requirements for a specific technical area, and *Design Teams (DTs) (part of the Software Unit)* that design, develop and maintain software for the previously defined functionality. Each RT has a team leader who manages the team. Another role belonging to the Requirements Unit is the *Requirements Architect* who is responsible for managing the scope at the high level and therefore also manages and coordinates the RTs. In the DTs there are a number of different roles, namely

- *Design Team Leader* who leads and plans the team's work for the implementation and maintenance phase
- *Design Team Requirements Coordinator* who leads the team's work during the requirements management and design phase, and coordinates the requirements with the RTs
- *Developer* who designs, develops and maintains the software
- *Tester* who verifies the software

The software unit also has a project management team consisting of among others *Quality Managers* who set the target quality levels and follow up on these*, and Software Project Managers* that monitor and coordinate the DTs and interact with the Requirements Architects. The product development unit is responsible for a number of different tasks, for this study *System Testing* is relevant. By testing

the products, including the platform they have experience of the implemented functionality and quality of the platform.

The company uses a stage-gate model with several increments. There are *Milestones (MSs)* and *Tollgates (TGs) for controlling and monitoring* the project progress. In particular, there are four milestones for the requirements management and design before the implementation starts: MS1, MS2, MS3, and MS4, and three milestones for the implementation and maintenance: MS5, MS6, MS7. For each of these milestones, the project scope is updated and baselined. The milestone criteria are as follows:

MS1: At the beginning of each project, long-term RT roadmap documents are extracted to formulate a set of features for an upcoming platform project. A *feature* in this case is a concept of grouping requirements that constitute a new functional enhancement to the platform. At this stage, the features usually contain a description, its market value and effort estimates. The level of details for the features should be set up in a way that enables judgment of its market value and implementation effort. Both values are obtained using a cost-value approach [16]. The cost for implementation and the market value of features are the basis for initial scoping inclusion for each technical area. The features are reviewed, prioritized and approved. The initial scope is decided and baselined per RT, guided by a project directive and based on initial resource estimates in the primary receiving DT. The scope is then maintained in a document called *Feature List* that is regularly updated each week after a meeting of the *Change Control Board (CCB).* The role of the CCB is to decide upon adding or removing features according to changes that happen. **MS2:** Features are refined to requirements which are specified by the RTs. One feature usually contains ten or more requirements. The features are assigned to main DTs that are responsible for designing and implementing the assigned features. The requirements are reviewed together with the main DTs and approved. Other (secondary) DTs that are also affected by the features are identified. The DTs make an effort estimate per feature for both main and secondary DT. **MS3:** DTs refine system requirements and start designing the system. The set of secondary DTs are refined along with the effort estimates, and the scope is updated and baselined. **MS4:** The requirements refinement work and the system design are finished, and implementation plans are produced. The final scope is decided and agreed with the development resources, i.e. the software unit. **MS5:** All requirements are developed and delivered to the platform. **MS6:** The software in the platform is stabilized and prepared for customer testing. **MS7:** Customer-reported issues are handled and the software updated. The software is ready to be released.

According to the company process guidelines, most of the scoping work should be done before reaching the second milestone of the process. The requirements are written in domain-specific, natural language, and contain many special terms that require contextual knowledge to be understood. In the early phases, requirements contain a customer-oriented description while being refined to detailed implementation requirements at a late stage.

IV. Research Methodology

The research was conducted using a two-phase qualitative research approach. Qualitative research aims to understand complex phenomena in the context where they exist. It can also be considered to be a suitable design for an improved understanding for a phenomenon of which the understanding is limited [30]. The phenomena can be investigated by studying peoples' verbalized thoughts about the context in which they act [20]. A qualitative approach is suitable when the study focuses on diversities rather than similarities. It is also appropriate when individual perceptions are to be studied prospectively, using a series of interviews [27].

The study reported in this paper has been planned into two phases. In the first phase one of the authors, with experience from working for the case company, analyzed the current situation within the company and derived a set of hypotheses that describes the causes and challenges related to requirements engineering (of which overscoping was one). The results from phase one were used in the second phase of the study, namely the interview study with practitioners in the case company. The details of both phases are described in the sections that follow.

A. *Phase one – an investigation of the situation at the company conducted by an experienced practitioner.*

Based on the experience of one of the authors (that has been an employee at the case company, with experience in several areas including coding, design, requirements engineering and process development), a number of challenges around requirements engineering in industry were identified, as well as, a number of possible causes and effects of these challenges. In this paper, we report only on the causes and effects related to one of the challenges, namely overscoping. These were discussed and adjusted over a period of a couple of months and they served as input for creating an interview instrument for phase two of the study. For overscoping, the following possible causes were identified:

1) Continuous requirements inflow via multiple channels (C1) The continuous inflow of requirements from the market and internally is managed by batching those requests into one or two projects per year. In addition to new requests via these projects, a lot of change requests come in after the scope for a project is set.

2) No consolidated overview of software development resources (C2) The resource allocation for software development resources is handled at the DT level, i.e. each DT manages the allocation of resources to different projects. This means that there is no consolidated view of the current load and/or availability of the resources for the Software

Unit as a whole either in total or per project, since each DT has that information in their separate plans.

3) Lack of DT involvement in early phases (C3) DTs are not involved enough in the early project phases (MS1-MS4) with providing cost estimates and feedback during requirements writing.

4) Requirements not agreed with DT (C4) The requirements specification is not always agreed with DTs at MS2. Even if there is a formal review by DTs, it is often with a low level of commitment from DTs both with respect to agreement on requirement details and feasibility of the suggested scope.

5) Detailed requirements upfront (C5) A detailed requirements specification (SRS) is produced by the RTs by MS2. Setting the detailed requirements before the design starts limits the room for negotiation around details that could enable a more efficient design and realistic development plan, as well as, allowing for a more agile handling of requirement changes caused by shifting market needs.

B. Phase two – an interview study with other practitioners in the case company

The purpose of the interview instrument is to explore and challenge the conjectures about reasons of overscoping in terms of both omissions and commissions, as well as, to investigate the consequences of the explored reasons. A set of additional requirements-engineering challenges were also included in the interview instrument, but not analyzed in this paper. Interviews were planned to be semi-structured with a high degree of discussion between the interviewer and the interviewee. All challenges planned to be discussed during the interviews were considered as complex challenges that can have many facets. Thus, for each of the main challenges (including overscoping) an open ended question about the challenge was asked: if it was a challenge, what causes it and what effects it has. This was done to find the root causes of the main challenges without imposing the assumptions made during the pre-study on the interviewee. If the interviewee did not explicitly mention an assumed cause they were specifically asked about their view on it. This step was performed to find the root causes of challenges derived from the pre-study and also issues added during the interview [30]. A section at the beginning of the interview was designed to briefly go through the development process and allow the interviewee to talk freely about the roles and phases she had experience from. The interviews were scheduled for 90 minutes each with the possibility to reduce time or prolong it. It was also decided to record all interviews for further transcription and analysis. After the interviews, transcripts were planned to be sent back to the interviewees for validation. Since we wanted to cover the whole chain from requirements definition through development (design, implementation and testing) to the end product (quality assurance, product projects) people from all relevant organizational units (Requirements, Software and Product development, see Section III) were selected. Nine persons were selected to be interviewed. Two of the interviewees (with identical roles) requested to have their interview together. The roles and organizational belonging, as well as, the experience of the interviewed persons can be found in Table I. We have used a coding for the interviewees that also includes their organizational belonging. For example, interviewees belonging to the requirements unit are tagged with a letter R, belonging to product unit with a letter P and belonging to software development unit with a letter S.

TABLE I. INTERVIEWEES ROLES, ORGANIZATIONAL BELONGING AND EXPERIENCE (SEE SECTION III FOR ROLE DESCRIPTIONS)

Code	Organizational unit	Role (experience)
Ra	Requirements	RT leader (5 years)
Rb	Requirements	RT leader (2 years)
Rc	Requirements	Req Architect (3 years)
Pd	Product	System Test Manager (7 years)
Se	Software	Tester (3 years)
Sf	Software	Software Project Manager (2 y), DT leader (2 y), Developer (2 y)
Sg	Software	Quality Manager (3 years)
Sh	Software	DT reqs coordinator (0,5 y), Developer (2 y), DT leader (1 year)
Si	Software	DT reqs coordinator (7 years)

C. Threats to validity

The quality of a qualitative study relies on the quality of the investigator [27]. All persons involved in this interview study have previous experience in conducting interview studies. The interview instrument used in this study is a result of a pre-study where the goal of the study and the questions to ask during the interviews were discussed and improved iteratively. During the analysis phase, there is a risk of quotations becoming out of context when dividing into separate coded segments [7]. To address this threat the observer triangulation method was used [27]. One researcher randomly selected two interview recordings and performed an independent transcription and coding. Differences were then discussed and conflicts were resolved. Furthermore, we have used a data triangulation method where an analysis of the current situation derived by an observation inputs and complements the interviews conducted [27].

V. RESULTS

The analysis performed in phase one of this study revealed that the overscoping challenge is a complex phenomenon which can have many potential causes. The results of the interviews presented in this section are divided into two main parts. The first part comprises the results of validating the five assumed causes, derived from phase one of the study, while the second part of this section comprises the results of the follow up questions about the root causes of the five main causes discussed during the interviews,

complemented by the interviewees' views on the effects of overscoping.

While analyzing the results, a sixth main cause for overscoping was identified. Five of the eight interviewees mentioned issues related to company-wide strategy and plan for future products and weak business priority of scope. Based on that the following cause has been added: *Unclear vision of overall goal (C6)*. The overall strategy concerning business directions for software development is unclear. Instead scope is proposed from a technology aspect for each area. The result is a prioritized list, where almost everything is 'Critical' and there is no unified priority of the whole scope for a project.

A. Validation of Assumed Causes (C1-C5)

The viewpoint of each interviewee was categorised per organization and matched against the original assumptions about causes for overscoping, as well as, the added cause concerning unclear vision of overall goal. The results are presented in Table II where the viewpoint for each interviewee on the impact of each cause on overscoping (the main challenge) was classified in the following way:

Experienced: the cause, both its occurrence and its impact on challenge, is experienced and was mentioned without prompting

Agreed: either the cause (occurrence and impact) was not directly mentioned, but derived or agreed after direct question, or when interviewee has no direct experience, but observed it/heard about it from others

Partly agreed: partly Experienced or partly Agreed

Disagreed: does not agree to the cause, either its occurrence or that it contributed to the main challenge

Not mentioned: even though within expected experience for role

NA: not within the expected experience for the role (according to the process).

TABLE II - RESULTS FOR CAUSES OF OVERSCOPING PER ORGANIZATIONAL UNIT

	Overall			C1 Cont req inflow			C2 No consolidtd view of software capacity			C3 Lack of DT involvement			C4 Reqs not agreed			C5 Detailed reqs upfront			C6 Weak vision of overall goal		
Organizational unit	Reqs	Software	Product	Reqs	Software	Product	Reqs	Software	Product	Reqs	Software	Product	Reqs	Software	Product	Reqs	Software	Product	Reqs	Software	Product
Experienced	2	5	1	1	3	1	1	2		3	1		1	1			1		3	2	
Agreed	1			2			2				1			1		1	1				
Partly agreed					1			1			2			2		1					
Disagreed																					
Not mentioned													2			1	2				
NA					1			2	1		1	1		1	1		1	1			

All of the interviewees had *Experienced* or *Agreed* to overscoping as a challenge. A majority of the interviewees had *Experienced* or *Agreed* to causes 1-3, while causes 4 and 5 both had less than a majority *Experienced* or *Agreed* and some *Not mentioned*. See Table II for more details. For all pre-assumed causes there were some counts of *Partly Agreed*.

Overscoping in general The two RT leaders (Ra and Rb, who were interviewed together) had different opinions about to which extent overscoping was a challenge. They had both experienced it as a challenge, but one of them (Rb) saw it as a manageable challenge and was therefore classified as 'Agreed'.

Continuous requirements inflow (C1): The Quality Manager interviewee (Sg) is classified as 'Partly Agreed' since she did not mentioned the aspect of parallel projects contributing to overscoping, just that there was a continuous inflow of requirements changes after the scope was set and that this contributed to overscoping.

No consolidated view of Software Unit capacity (C2) Interviewee Si agreed to the fact that there was no consolidated view of the available resources and that this increased the overscoping, though the interviewee did not believe it would have alleviated the problem to any great extent, indicating that the cause of overscoping was caused to a lesser degree by the lack of consolidated information on available software development resources. On the other hand, interviewee Sf saw this as a strong cause for the overscoping; "*There was no plan. There was no control of what people were working with. There were different DT leaders who just pumped out [tasks]*."

Lack of DT involvement in early phases (C3) Both the DT requirements coordinators (Sh, Si) agreed to the lack of cost estimates from the DTs, though they both stated that they in their role as DT requirements coordinators have experienced a good cooperation with the RT leaders during the phase where the system requirements specification is produced (MS1 – MS2.)

Requirements not agreed with DT (C4) Similarly to the previous cause (Lack of DT involvement) the DT requirements coordinators (Sh, Si) agreed that the requirements were not committed to by the development team at MS2, but they experienced that the requirements themselves were agreed. One of the DT requirements coordinators (Sh) mentioned that the agreement around the system requirements specification was primarily with the DT requirements coordinators and not with the developers and testers in the DT.

Detailed Requirement Specification produced upfront (C5) The validation of this cause is weaker than for the others, since three of the interviewees are noted as *Not mentioned*. The only interviewee who *Experienced* that C5 contributes to overscoping was Sh. The interviewee stated that it was easy for the DTs to get enthusiastic when seeing the detailed requirements and committing to features without having sufficient resources to complete them. Interviewee Rb (RT leader) was classified as *Agreed*. This responder has an agile way of working where the requirements are not formally documented. Instead a significant amount of time is spent on regularly and directly interacting with the DT. The interviewee's experience is that this gives the RT leader more control and insight into the actual development and enables a more flexible discussion around detailed requirements, making the challenge of overscoping more manageable. Interviewee Ra (who was interviewed at the same time) agreed to Rb's conclusions and was classified as

Partly Agreed. Interviewee Sf believed that the wasted effort of producing and agreeing to detailed requirements for features that were later descoped increased the overscoping since it hindered those resources from working on viable features.

The entries marked *NA* (Not Applicable) indicate that interviewee is not expected to have experience of an item as causing the main challenge. The System Test Manager (Pd) and the Quality Assurance Manager (Sg) were both classified as *NA* for all causes except C1 (Continuous inflow) since they in their roles are not involved in the early phases of the projects or in the requirements work, while still having insight into the requirements flow from their positions at the management level. The SW tester (Se) was not involved in the planning and work distribution, and the causes C1 (Continuous inflow) and C2 (No consolidated overview of SW capacity were marked as *NA* for her.

B. Root Cause Analysis of Causes of Overscoping

To provide a deeper understanding, rather than just confirming or rejecting our assumptions about the causes of overscoping derived from the pre-study, the interviewees were asked to describe the root causes that may be triggering overscoping for each of the assumed causes (C1 to C5.) The additional triggers mentioned during the interviews (resulting in identifying the sixth cause C6) are included as root causes for C6. In Figure 1, the full picture summarizes our interpretation of the interview material concerning the root causes of the six main causes (outlined in Sections IV.A and V). For each root cause and effect the interviewee from whom the connections are derived is noted within parentheses.

Root causes of C1: Continuous requirements inflow via multiple channels

a) Long lead times (Rc)
b) Large number of product variants (Rc. Sf)
c) Software-internal roadmap (Ra, Sf)
d) Indirect requirements on other DTs (Sf, Sh)

In addition to the regular requirements flow, defined by the RTs, two other sources of requirements were mentioned; the Software Unit itself (Ra, Rb and Sf) and the products in the product line that required new variants to be supported (Sf, Rc.) One of the DT requirements coordinators (Sh) also described that as the design work progressed (after MS2), an indirect requirements flow was identified with requirements on other than the main DTs. These requirements, caused by dependencies in the system, greatly increased the amount of requirements to be implemented, especially for DTs responsible for service-layer functionality like UI framework and communication protocols. The long lead times of the regular requirements flow (approx 2 years) caused (as seen by Rc) market and product requirements coming in later and both causing changes to the scope and also increasing the overscoping.

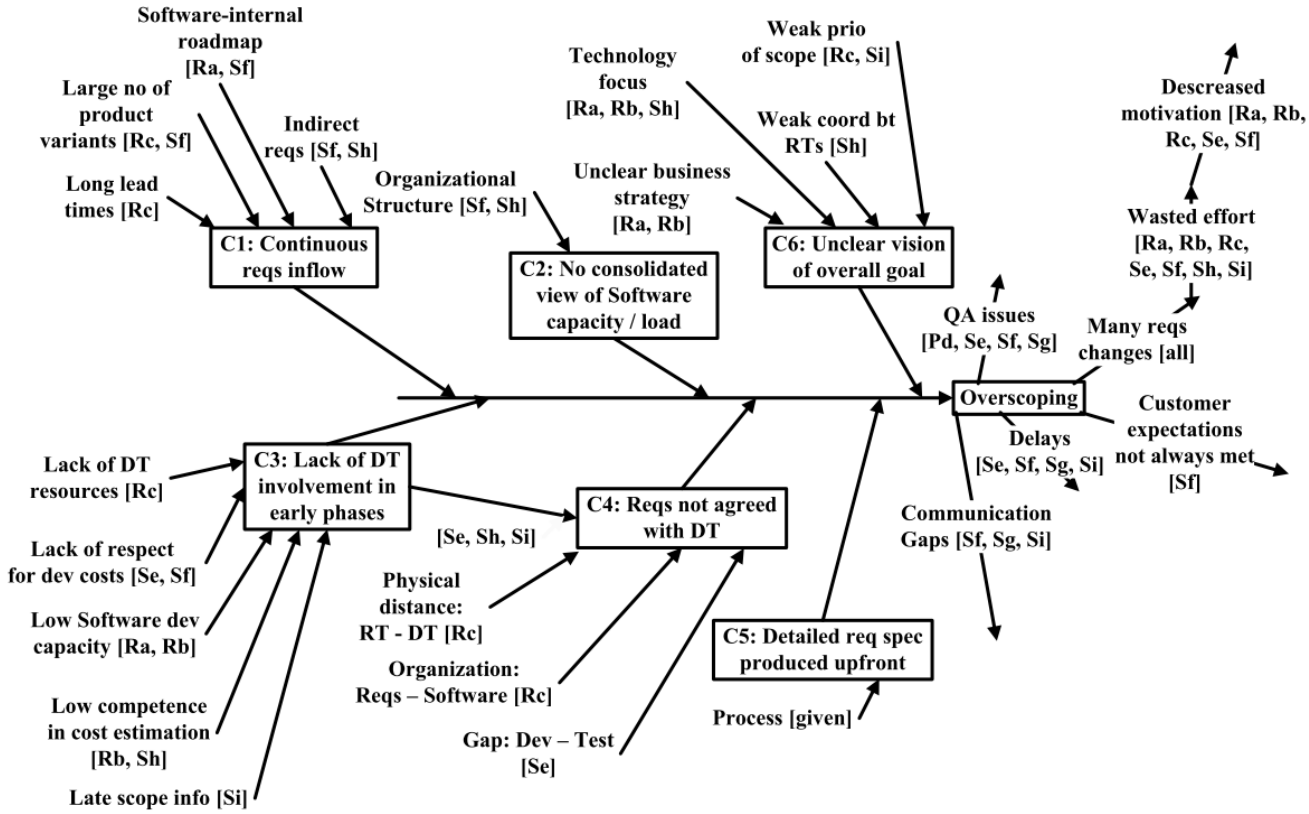

Figure 1. Causes, root causes and effects of overscoping. Reference to interviewee code noted within brackets.

Root causes of C2: No consolidated view of Software Unit capacity

a) Organizational structure (Sf, Sh)

Two out of three interviewees with experience from DTs clearly mentioned the organizational structure as the cause of not having a consolidated view of availability of the software development resources. Other parts of the interview material show that there are communication problems between DTs.

Root causes of C3: Lack of DT involvement in early phases

a) Lack of DT resources for pre-development work (Rc)
b) Lack of respect/understanding for development costs (Se, Sf)
c) Low Software Unit development capacity (Ra, Rb)
d) Low competence in cost estimation (Rb, Sh)
e) Late scope info to DT (Si)

Concerning having DTs involved in the early phases of the projects, interviewee Rc says: *"That was what we tried to do by MS2, but it was hard to get [DT] resources. To be honest, that probably was the problem."* In contrast to this, one DT reqs coordinator (Si) mentioned that at times the DT would get information on the new requirements very late, not giving the team sufficient time to analyse them. This was thought to be an effect of the RT and DT belonging to different organizational units. The difficulty in making good cost estimates was mentioned by Rb and Sh as a reason for the estimates (that were made at MS2 and later) not correctly reflecting the actual development effort. The two RT leaders (Ra, Rb) strongly expressed that the Software Unit's development capacity was very low and believed this was due to bad architecture. On the other hand, two interviewees from DTs (Se, Sf) described situations where cost estimates for both development and for testing were not respected and decisions to accept requirements for implementation were made without realistic plans.

Root causes of C4: Requirements not agreed with DT

a) C3 Lack of DT involvement in earlier phases (Se, Sh, Si)
b) Physically distance between Requirements and Software units (Rc)
c) Organizational structure, separate Requirements and Software units (Rc)
d) Communication gap between Developers and Testers within DT (Se)

All three interviewees with experience from planning DT development (Se, Sh, Si) stated that the lack of DT involvement in the early phases (C3) caused a lack of commitment to develop the new requirements. The view on the level of agreement to the actual detailed requirements varied between interviewees Si and Sh, see validation of C3 and C4 in Section V.A. Interviewee Si experienced a good cooperation with the RT, but that the different organizational belongings caused timing issues due to different priorities and focus of the different units. One interviewee (Rc) had experienced that, in general, the requirements were not agreed between RT and DT, and believed that this was due to weak communication caused by organizational and physical distance (office space in different buildings) the teams. The interviewed DT tester (Se) had also experienced weak communication between developers and testers around both requirements and design within the DT that caused lack of agreement.

Root causes of C5: Detailed requirements specification

a) Given by the process

The process defines that a requirements specification should be produced by MS2.

Root causes of C6: Unclear vision of overall goal

a) Unclear business strategy for software development (Ra, Rb)
b) Scope set from technology focus (Ra, Rb, Sh)
c) Weak coordination between RTs (Sh)
d) Weak priority of scope (Rc, Si)

Four of the interviewees (Ra, Rb, Rc, Sh) mentioned these related issues as causes for overscoping. The RT leaders (Ra and Rb) lacked a clear business strategy and vision that could guide them in defining features for their technical area. Instead they, and interviewee Sh, stated that the project scope was proposed from a pure technology standpoint. Sh had also experienced that the scope defined by the RTs was not coordinated between RTs. Si described that due to a weak business priority of the features proposed by each RT (almost everything was 'critical') the DTs were pushed to commit to unrealistic project plans. And Rc saw the lack of a unified priority of the suggested features as a factor that made it hard to effectively address the challenge of having too much in scope at the project level.

C. Effects of Overscoping

Five main effects of overscoping were identified in the study. They are as follows:

E1: Many changes after project scope is set (all) which in turn has the effect of

- Wasted effort (Ra, Rb, Rc, Se,Sf, Sh, Si)
- Decreased staff motivation (Ra, Rb, Rc, Se, Sf)

All interviewees have experienced that a large amount of requirement changes take place after the project scope is (supposed to be) set, i.e. at MS2. It is very common for large amounts of features to be removed from scope (descoped) as the project proceeds. The phenomena is so common that the phrases 'overscoping' and 'descoping' have become part of company vocabulary. Five out of the eight interviewees expressed frustration and decreased motivation, mainly to work with new requirements, as an effect of overscoping since both the requirements and the design work (and, in some cases, implementation work) is wasted when features are removed from the project scope. One of the interviewees with experience from working in a DT said: "*There are many things that you as a tester or developer have spent time on that never resulted in anything. And that isn't very fun. There is a lot of overtime that has been wasted."* [Sh] The effect was not as serious for the system testing at the Product Unit; they had to adjust their test plans accordingly.

E2: Quality issues (all)

An effect of overscoping that was mentioned by Pd, Se, Sf and Sg, i.e. all the roles involved in the late phases of the

project, was quality issues. The many requirements changes (E1) were seen as contributing to the quality problems. Interviewee Pd said: *"When you have a lot going on at the same time everything isn't finished at the same time and you get a product with lower quality."* Another factor that was seen to contribute to the quality issue by the software tester (Se) was *lack of DT input in early phases* (C3) and for her the main experienced root cause was 3b) *Lack of respect for estimates on development*, and in this case, cost for testing.

E3: Delays (Se, Sf, Sg, Si)

Several of the interviewees mentioned delays as being a common effect of having too much in scope, in combination with the many changes. The Quality Manager (Sg) said: *"Even if we decide on a scope for MS4, there are extremely many changes underway so we are never ready by MS5 as we have said, but it is delayed since there are so many new things that appear."*

E4: Communication gaps (Sf, Sg, Si)

As well as, being a cause for overscoping, communication gaps were seen as an effect of pushing too much work on to the software unit. These gaps were mentioned both between the requirements and software units, between the software and the product unit, but also within the software unit. For example, the product unit test according to the requirements specification, but due to all the changes (partly caused by overscoping) and lack of communication of them to the product unit, a large number of false error reports are filed on incorrect (not updated) requirements. The software-internal roadmap is another example of where the communication gap caused by overscoping has resulted in the software unit driving new requirements themselves without coordination with the requirements unit.

E5: Customer expectations are not always met (Sf)

One interviewee (Sf) mentioned that the increased set of requirements caused by the large number of different products that the product line is to support, are not always correctly implemented, resulting in problems with the user interface, e.g. icon placed a few pixels out of place.

VI. Interpretation and Discussion

In this section, we provide our interpretation and discussion of achieved results, concluded by discussing the limitations of this study. In general, we interpret overscoping to be a challenge for the case company caused by a number of different factors mainly originating from the nature of a market-driven business domain, but also due to communication gaps and organizational structures within the company. The study identified that the scope is mainly set from a technology focus, rather than a business perspective, due to lack of strategy for software development. The effects of overscoping are serious, according to the interviewees. The most serious one being the impact it has on the quality of the end product. Delays and quality problems are expensive, not just considering the cost of fixing the quality issues, but also in reduced market shares, decreased brand value etc. Overscoping also results in effort and time being wasted on features that are started, but never completed due to being descoped, that in turn leads to a long term effect of reduced staff motivation. The communication gap between the Requirements Unit and the Software Unit causes a lot of problems, among others that the Software Unit sees a need to define their own roadmap in addition to the roadmap of the Requirements Unit and spend resources on developing functionality in parallel to the development projects. The contrast between the RT leaders' opinion that Software Unit capacity is very low and the experience from the DT that development cost efforts were not respected is very interesting and at least shows that there is a lack in understanding each other's viewpoint around cost. If the development capacity really is low is a different issue.

The nature of the business domain with a lot of changes and high market pressure requires an organizational set-up and process that is suitable for handling change in an efficient and cost effective way. The current process has been designed to handle change, but does not do so sufficiently well. It needs to be more agile, while at the same time handle the complexities of developing embedded software in a large setting. As one interviewee stated: *"The waterfall approach is good from a preparation perspective, if you can then stick to what is planned, but since we live in a world that changes a lot it doesn't work after all."*

As for every study there are limitations that should be discussed or addressed. In our case, we discuss limitations of the research design in Section II, while in this section we discuss other limitation and threats to validity, based on guidelines provided by Robson [27]. The main threat to description validity is to provide a valid description of what interviewees said and meant. This threat was addressed by making audio recordings of all interviews and transcribing them. Before analyzing the full material the transcripts will be sent back to the interviewees to check for misinterpretations and other errors. Considering the treats to valid interpretation, the main issue has been not to impose the understanding of the main challenges and their causes and effects derived from the pre-study on the interviewees. To address this threat, the question on each challenge (of which overscoping was one) were formulated in an open and indirect way so that the interviewee could have a chance to express her own opinion about a certain fact without immediately focusing on judging if a mentioned cause is relevant or not. Also, the open questions allowed identification of the sixth cause which confirms that our pre-defined categories were subjected to checking of their appropriateness and extended. To enhance *reliability* of the results by minimizing the number of transcription errors, the person transcribing all the interviews was always present at the interviews and taking notes during the interview. Moreover, this person worked for the case company at the time of the study (and has done so for several years) which means that the person has extensive knowledge and experience about the company culture and language, enabling the richness and the level of detail resulting from the study. A possible source of unreliability is related to

observer biases where the results from the pre-study, as well as, questions asked during the interview, may have been consciously or unconsciously biased by the practicing researcher's own knowledge of the people, processes and problem. This threat to validity was addressed during a set of meetings with other researchers involved in this study to discuss findings from the pre-study and decide on the best possible set of people to interview. The interview instrument was also reviewed and agreed in this group, in order to avoid imposing assumptions derived from the pre-study on the interviewees. Moreover, this practitioner's involvement in the study has played a vital role in ensuring that the problem under investigation is an authentic problem, that the interpretation of data is based on a deep understanding of the case and its context, and that the outcome of the study is authentic. Finally, the results from the interviews confirm the main part of the assumptions resulting from the pre-study. The main threat to *theory validity* was addressed by seeking data which is not confirming a pre-assumed theory and reporting differences and disagreements in Section V.

Finally, the possibility of generalizing the results of this case study has been addressed both internally within the study and in respect to external generalisability. The *internal generalisability* was addressed by sampling interviewees from different parts of the company having different roles and responsibilities. As for *external generalisability* of this case study the main threat to validity is no possibility of performing a statistical generalization due to lack of representative sample and only one company involved in the study. However, the main focus on this study is to increase the understanding of scoping activity and explore possible causes of overscoping rather than providing a full theory and all possible causes and reasons for overscoping that can be generally applied. Even though the nature of qualitative studies makes it practically impossible to perform an exact replication, the follow up studies in other domain can help in understanding other cases and situations. Moreover overscoping was confirmed as a challenge by all our responders with only minor differences of the importance of this issue. Finally, some of the causes of overscoping outlined in Section V.A have also been reported by other studies [17][24].

VII. Conclusions and Further Work

Release planning, as a part of release management [32], is one of the most important and challenging [16] tasks in MDRE, since it deals with the essential question of what to develop and when, which is inherently related to the challenge of meeting customers' demands [2]. While doing release planning, a Software Product Manager has to look ahead and plan the content of several software product releases [13] and provide right products at the right time to users. Even though much research has been carried out in the area of release planning, including various techniques to assist in selecting which requirements to include and exclude from the scope of the upcoming release [16][18][24], other core functions of this process, like scope management have been less explored [32]. Moreover, many of the techniques proposed and used to prioritize requirements [1][16][18] focus on setting the scope of the next project as a discrete activity, or in the series of releases [12], while our previous work reported that scoping can be a continuous activity that can last throughout the entire project lifecycle [33]. In addition, many of the prioritization techniques assume that the set of candidates to select from is known 'a priori' which in a MDRE context with a constant flow of incoming requirements is hard to ensure [24]. The result may be a situation of overscoping when more requirements are included into the scope than the project can handle.

In this paper, we report on an interview study that aims to understand causes and effects of overscoping. The interview study has been preceded by a pre-study at the case company that identified possible causes of overscoping, as well as, effects of overscoping. The results from the pre-study were discussed, extended and validated during nine in-depth interviews with practitioners working with requirements engineering, software development and product testing. Our responders mostly confirmed five assumed causes of overscoping and stressed the need for a sixth cause of overscoping, namely unclear vision of overall goal. Additionally, this paper reports on root causes for overscoping and provides a list of five possible effects of overscoping mentioned by our responders.

Future work includes complementing the analysis of the full interview material on other challenges in requirements engineering, including their causes and root causes. It is also planned to analyze how different challenges are interconnected. Furthermore, we plan to investigate how aspects such as organizational set-up, software development model (agile or waterfall) and application of different software engineering methods affect the challenges and their causes and root causes both within the case company, and in a broader context. This requires widening the research to also include other software development companies.

Acknowledgment

We would like to thank all anonymous interviewees for their invaluable contribution to this project. The project is partly funded by the Swedish Foundation for Strategic Research and VINNOVA (The Swedish Governmental Agency for Innovation Systems) within the EASE project.

References

[1] M. Abramovici and O. C. Soeg, "Status and Development Trends of Product Lifecycle Management Systems", Ruhr-University Bochum, Germany, 2002.

[2] P. Carlshamre and B. Regnell, "Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes", Proc. 11th International Workshop on Database and Expert Systems

Applications, IEEE Computer Society, Sep. 2000, p. 961, doi:10.1109/DEXA.2000.875142.

[3] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell and J. Natt och Dag, "An industrial survey of requriements interdependencies in software product release planning" In Proc. of the Fifth IEEE International Symposium on Requirements Engineering (RE01), IEEE Computer Society , Aug. 2001, pp. 84–91, doi:10.1109/ISRE.2001.948547.

[4] P. Carlshamre, "A usability perspective on requirements engineering – From methodology to product development" Ph.D Thesis, Linköping University Sweden, 2002.

[5] P. Carlshamre, "Release planning in market-driven product development: Provoking an understanding", Requiremetns Engineering Journal, vol. 7, Apr. 2002, pp. 139-151, doi:10.1007/s007660200010.

[6] J. Chen, R. R. Reilly and G. S. Lynn, "The impacts of speed-to-market on new product success: The moderating effects of uncertainty", IEEE Transactions on Engineering Management, vol. 52, May 2005, pp. 199-212, doi:10.1109/TEM.2005.844926..

[7] A. J. Coffey and P. A. Atkinson, Making Sense of Qualitative Data: Complementary Research Strategies, Sage Publications, Inc, 1996.

[8] D. Condon, "Software Product Management", Aspatore, Books, Boston, 2002.

[9] M. A. Cusumano and R.W. Selby, "Microsoft Secrets", Simon and Schuster, New York, 1995.

[10] C. Ebert and J. De Man, "e-R&D – Effectively Managing Process Diversity", Annals of Software Engineering, vol. 14, , Dec. 2002, pp. 73-91, doi:10.1023/A:1020545406509.

[11] J.-M. DeBaud, K. Schmid, "A systematic approach to derive the scope of software product lines", Proceedings of the 21st International Conference on Software Engineering, *ACM*, Los Angeles USA, May 1999, pp. 34-43, doi:10.1145/302405.302409.

[12] D. Greer and G. Ruhe, "Software release planning: an evolutionary and iterative approach." Information and Software Technology, vol. 46, March 2004, pp. 243-253, doi:10.1.1.89.3772.

[13] A. Van der Hoek, "Software release management", Proc. Of the Sixth European Software Engineering Conference together with the Fifth ACM SIGSOFT Symposium on the Fundations of Software Engineering, Springer: Heidelberg, Germany, 1997, pp. 159–175, doi:10.1145/267896.267909.

[14] A. F. Hutchings and S. T. Knox, "Creating products customers demand", Communications of the ACM, vol. 38, May 1995 pp. 72-80, doi:10.1145/203356.203370.

[15] M. Jorgensen and M. Shepperd, "A Systematic Review of Software Development Cost Estimation Studies", IEEE Transactions on Software Engineering, vol. 33, January 2007, pp. 33-53, doi: 10.1109/TSE.2007.256943.

[16] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing requirements", IEEE Software, vol. 14, Sep. 1997, pp. 67–74, doi:10.1109/52.605933.

[17] L. Karlsson, Å. G. Dahlstedt, J. Natt Och Dag, B. Regnell and A. Persson, "Requirements engineering challenges in market-driven software development An interview study with practitioners", Information and Software Technilogy, vol. 49, June 2007, pp. 588-604, doi:10.1016/j.infsof.2007.02.008.

[18] L. Karlsson, T. Thelin, B. Regnell, P. Berander and C. Wohlin, " Pair-wise comparisons versus planning game partitioning - experiments on requirements prioritisation techniques", Empirical Software Engineering, vol. 12, Feb. 2007, doi: 10.1007/s10664-006-7240-4.

[19] M. Lubars , C. Potts and C. Richter, "A Review of the State of the Practice in Requirements Modeling", Proc. IEEE International Symposium on Requirements Engineering (RE93), IEEE Computer Society Press, pp. 2-14, Jan. 1993, doi: 10.1109/ISRE.1993.324842.

[20] M. D. Myers and D. Avison, "Qualitative Research in Information Systems", Sage, 2002.

[21] C. Pohl, G. Böckle, and F. J. van der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques,* Springer-Verlag, New York USA, 2005.

[22] C. Potts, "Invented requirements and imagined customers: requirements engineering for off-the-shelf software", Proc. of the Second IEEE International Symposium on Requirements Engineering, IEEE Comp. Soc. Press, March 1995, pp. 128-131, doi:10.1109/ISRE.1995.512553.

[23] B. Regnell, P. Beremark and O. Eklundh "A market-driven requirements engineering process – results from an industrial process improvement", Requirements Engineering Journal, vol. 3, Jun. 1998, Springer, pp. 121–129, doi: 10.1007/BF02919972.

[24] B. Regnell and S. Brinkkemper, "Engineering and Managing Software Requirements" chapter in "Market–Driven Requirements Engineering for Software" by C. Wohlin and A. Aurum , pages 287–308. Springer, 2005.

[25] P. Sawyer, "Packaged software: Challenges for re", Proc. 6th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'2000), Stockholm, Jun. 2000.

[26] P. Sawyer, I. Sommerville and G. Kotonya, "Improving Market-Driven RE Processes" In: Proceedings of the International Conference on Product-Focused Software Process Improvement (Profes 1999), Oulu, Finland, June 1999, pp. 222–236 .

[27] C. Robson. Real World Research. Blackwell Publishing, 2002.

[28] T.L. Saaty, "The Analytic Hierarchy Process" McGraw-Hill, New York, NY, 1980.

[29] K. Schmid, "A Comprehensive Product Line Scoping Approach and Its Validation", 24th International Conference on Software Engineering (ICSE 2002*), IEEE Computer Society*, Orlando USA, May 2002, pp. 593-603, doi: 10.1109/ICSE.2002.1008004.

[30] Strauss A, Corbin J. Basics of Qualitative Research: Grounded Theory Procedures and Techniques, Sage, 1990.

[31] S. Unger, "Ten marketing Challenges that Can Make or Break your Business… and How to Address Them", http://www.pragmaticmarketing.com/publications/magazine/1/1/01su, last accessed July 2010.

[32] I. Van De Weerd, S. Brinkkemper, R. Nieuwenhuis, J. Versendaal and L. Bijlsma, "A Reference Framework for Software Product Management", Technical Report UU-CS-2006-014, Department of Information and Computing Sciences Utrecht University, 2006, ISSN: 0924-3275.

[33] K. Wnuk, B. Regnell and L. Karlsson, "What Happened to Our Features? Visualization and Understanding of Scope Change Dynamics in a Large-Scale Industrial Setting", Proc of 17th IEEE International Requirements Engineering Conference, IEEE Computer Society Press, Sep. 2009, pp. 89-98, doi:10.1109/RE.2009.32

[34] C. Wohlin, X. Min and A. Magnus, "Reducing time to market through optimization with respect to soft factos", Proc of the 1995 Annual International Engineering and Management Conference, Jun. 1995, pp. 116-121, doi: 10.1109/IEMC.1995.523919.