Are You Biting Off More Than You Can Chew? A Case Study on Causes and Effects of Overscoping in Large-Scale Software Engineering

Bjarnason, Elizabeth; Wnuk, Krzysztof; Regnell, Björn

Published in:
Information and Software Technology

DOI:
10.1016/j.infsof.2012.04.006

2012

Information and Software Technology xxx (2012) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering

Elizabeth Bjarnason *, Krzysztof Wnuk, Björn Regnell

Department of Computer Science, Lund University, Lund, Sweden

ARTICLE INFO

Article history:
Received 31 January 2011
Received in revised form 27 April 2012
Accepted 29 April 2012
Available online xxxx

Keywords:
Requirements scoping
Empirical study
Software release planning
Case study
Agile requirements engineering

ABSTRACT

Context: 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 market-driven case, when only a few requirements are known a priori, the risk of overscoping may increase.
Objective: This paper reports on findings from a case study aimed at understanding overscoping in large-scale, market-driven software development projects, and how agile requirements engineering practices may affect this situation.
Method: Based on a hypothesis of which factors that may be involved in an overscoping situation, semi-structured interviews were performed with nine practitioners at a large, market-driven software company. The results from the interviews were validated by six (other) practitioners at the case company via a questionnaire.
Results: The results provide a detailed picture of overscoping as a phenomenon including a number of causes, root causes and effects, and indicate that overscoping is mainly caused by operating in a fast-moving market-driven domain and how this ever-changing inflow of requirements is managed. Weak awareness of overall goals, in combination with low development involvement in early phases, may contribute to 'biting off' more than a project can 'chew'. Furthermore, overscoping may lead to a number of potentially serious and expensive consequences, including quality issues, delays and failure to meet customer expectations. Finally, the study indicates that overscoping occurs also when applying agile requirements engineering practices, though the overload is more manageable and perceived to result in less wasted effort when applying a continuous scope prioritization, in combination with gradual requirements detailing and a close cooperation within cross-functional teams.
Conclusion: The results provide an increased understanding of scoping as a complex and continuous activity, including an analysis of the causes, effects, and a discussion on possible impact of agile requirements engineering practices to the issue of overscoping. The results presented in this paper can be used to identify potential factors to address in order to achieve a more realistic project scope.

1. Introduction

Maximizing the business value for a product and a set of available resources may sound like a simple task of selecting features according to the highest return of investment. However, in market-driven requirements engineering (MDRE) [38,59] software product managers face the challenge of managing continuously shifting market needs [1] with a large number of new and changing requirements [28] caused both by a capricious market situation [19] and by evolving technologies. In this situation, selecting which requirements to include into the next release of a software product (also called *scoping* [66] or *project scoping* [56]) is a complex and continuous task of assessing and re-assessing how these scoping changes impact the common code base of the software product line [54] on which those products are built [71]. This domain scoping is considered part of the product line scoping [66], which derives value from the opportunities to reuse functionality of the product line. These factors, combined with increased market competition and unpredictable market response to new products, force decision makers to continuously face the task of making and re-evaluating decisions in an ever evolving world [5].

Defining the scope of a product to fit a required schedule is a known risk in project management [13] and in our previous work [71] we found that the project scope at a large software company changed significantly throughout the entire project life cycle. These changes were partly due to *overscoping*, i.e. setting a scope that requires more resources than are available. Several researchers have focused on *scope creep* where the scope is increased by the developers, and highlighted this as a serious project risk [16,17,31]. Others have investigated scoping as a part of release planning

* Corresponding author. Tel.: +46 46 222 00 00.
E-mail addresses: Elizabeth.Bjarnason@cs.lth.se (E. Bjarnason), Krzysztof.Wnuk@cs.lth.se (K. Wnuk), Bjorn.Regnell@cs.lth.se (B. Regnell).

0950-5849/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.infsof.2012.04.006

[66,68,71]. However, no study has yet attempted to investigate the causes and effects of overscoping even though requirements engineering (RE) decision making is an acknowledged challenge [2,5,50]. In this study, we have investigated this phenomenon of *overscoping* a project, or biting off more that you can chew, in particular in a market-driven and very-large scale RE (VLSRE) context [58].

Agile development processes claim to address several of the challenges involved in scoping frequently changing requirements. For example, in eXtreme programming (XP) [7] and Scrum [67] the balance between scope and available resources is managed by extreme prioritization and constant (re)planning of the scope in combination with time boxing of the individual development iterations. However, agile requirements engineering (RE) practices have also been found to pose new challenges, e.g., in achieving consensus on priorities among multiple stakeholders and in creating accurate project plans (cost and timeline) for an entire project [57].

The main goal of the case study reported on in this paper was to increase the understanding of factors involved in overscoping and thereby highlight this risk and take a step towards addressing and avoiding overscoping of projects. To achieve this, the study was designed to answer the following questions: (RQ1) what causes overscoping? (RQ2) what are the resulting effects of overscoping? and (RQ3) how may agile RE practices impact the causes and effects of overscoping? The case study has been conducted at a large market-driven software development company that has started to shift towards a more agile way of working. The study includes interviews with nine practitioners working with requirements engineering, software development and product testing. The interview results were then validated via a questionnaire with another six practitioners from the case company. The contribution of the presented work includes eight main causes of overscoping complemented by a number of root causes, and nine main effects of overscoping. In addition, the results indicate that three of the agile RE practices adopted by the case company may impact some of these causes and root causes and, thus, may also reduce the effects of overscoping.

Partial results from this study have previously been published as workshop publications in [11] where overscoping was preliminarily investigated and in [10] where preliminary results on the benefits and side effects of agile RE practices were published. For this article, the results are extended with (1) additional causes, root causes and effects of overscoping; (2) additional empirical results on overscoping from 6 (other) practitioners; and (3) details on the impact of agile RE practices specifically on overscoping. These extensions were achieved by further analysis of the full interview material and further validation of the results through a questionnaire.

The remainder of this paper is structured as follows: Section 2 describes related work. Section 3 describes the case company, while the research method is outlined in Section 4. The results are reported in Section 5 for the interviews and in Section 6 for the validation questionnaire. In Section 7, the results are interpreted and related to other work, and limitations and validity threats are discussed. Finally, Section 8 contains conclusions and further work.

2. Related work

Unrealistic schedules and budgets are among the top ten risks in software engineering [13] and some reasons for overloading projects with scope have been suggested. For example, DeMarco and Lister mentioned that a failure among stakeholders to concur on project goals [20] (also one of the challenges of agile RE [57]) can result in an excessive scope burden on a project. Project overload may also result from sales staff agreeing to deliver unrealistically large features without considering scheduling implications [29]. Furthermore, scope that is extended beyond the formal requirements by the developers, i.e. scope creep, is stated by Iaconou and Dexter [31] as a factors leading to project failures. Scope creep is also mentioned as having a big impact on risk and risk management in enterprise data warehouse projects [43]. In addition, it is listed as one of five core risks during the requirements phase, and is a direct consequence of how requirements are gathered [20]. On the other hand, Gemmer argues that people's perceptions of risk and their subsequent behaviour is overlooked within risk management and that an increased awareness of causes and effects of risks may lead to an improved discussion and management of these risks [26]. Some methods and tools to mitigate and manage risks related to scoping have been presented [17]. For example, Carter et al. [16] suggested combining evolutionary prototyping and risk-mitigation strategy to mitigate the negative effects of scope creep. However, the full issue of overscoping is not explicitly named as a risk in the related work, nor empirically investigated for their causes and consequences.

Requirements engineering (RE) is a decision intense part of the software engineering process [5], which can support and increase the probability of success in the development process [4]. However, the efficiency and effectiveness of RE decision making has cognitive limitations [5], due to being a knowledge intensive activity. Furthermore, research into the field of RE decision making is still in its infancy [2,50]. A major challenge in this research (according to Alenljung and Persson) lies in understanding the nature of RE decision making and identifying its obstacles [2] and several authors [2,4,5,50] mention the need to: (1) identify decision problems in the RE process; (2) perform empirical studies of RE decision making; and (3) examine how non-technical issues affect or influence decision making. Communication gaps are an example of such non-technical issues which have been reported to negatively affect the decision making and contribute to defining an unrealistic scope [12].

There are two characteristics of MDRE [59] which further aggravates RE decision making, namely a lack of actual customers with which to negotiate requirements [37,55] and a continuous inflow of requirements from multiple channels [28,38]. As a result, rather than negotiate with specific customers, the demands and requirements of an anonymous consumer market have to be 'invented' [55] through market research. Moreover, the success of the final product is primarily validated by the market with the uncertainty [59] of how well the 'invented' requirements compare to the market needs. Commonly, market research continuously issues more requirements [59] than can be handled with available resources. A state of congestion then occurs in the MDRE process [38] and the set of requirements to implement in the next project has to be selected using prioritization techniques based on market predictions and effort estimates [15,33,37].

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 [66]. Accurate release planning is vital for launching products within the optimal market window. And, this is a critical success factor for products in the MDRE domain [64]. Missing this window might cause both losses in sales and, additional cost for prolonged development and delayed promotion campaigns. However, making reliable release plans based on uncertain estimates [38] and frequently with features with dependencies to other features [14] is a challenge in itself. In addition, a rapidly changing market situation may force a project to consider new market requirements at a late project stage. Release planning is then a compromise where already committed features may need to be sacrificed at the expense of wasted effort [71] of work already performed. The area of release planning

Please cite this article in press as: E. Bjarnason et al., Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering, Inform. Softw. Technol. (2012), http://dx.doi.org/10.1016/j.infsof.2012.04.006

is well researched and Svahnberg et al. reported on 24 strategic release planning models presented in academic papers intended for market-driven software development [68]. Furthermore, Wohlin and Aurum investigated what is important when deciding to include a software requirement in a project or release [73]. Despite this, the understanding of the challenges related to scope management and their causes and effects is still low.

Scoping in agile development projects mainly involves three of the agile RE practices identified by Ramesh et al., namely *extreme prioritization*, *constant planning* and *iterative RE* [57]. High-level requirements are prioritized and the features with the highest market value are developed first. This approach ensures that if the project is delayed launch may still be possible since the most business-critical requirements will already be developed. Ramesh et al. identified a number of benefits for companies applying these agile RE practices, but also challenges and varying impact on project risks. The identified benefits include an ability to adapt to changing prioritization of requirements, as well as, a clearer understanding of what the customers want, thus reducing the need for major changes [57]. On the other hand, agile RE practices were found to include challenges in (1) correctly estimating and scheduling for the full project scope (which continuously changes), (2) a tendency to omit quality requirements and architectural issues (with the risk of serious and costly problems over time), and (3) constant reprioritization of the requirements (with subsequent instability and waste) [57].

3. The case company

The case company has around 5000 employees and develops embedded systems for a global market using a product line approach [54]. The projects in focus for this case study are technology investments into an evolving common code base of a product line (a.k.a. platform) on which multiple products are based. There are several consecutive releases of this platform where each release is the basis for one or more products. The products mainly reuse the platform's functionality and qualities, and contain very little product-specific software. A major platform release has a lead time of approximately 2 years from start to launch, and is focused on functionality growth and quality enhancements for a product portfolio. For such projects typically around 60–80 new features are added, for which approximately 700–1000 system requirements are produced. These requirements are then implemented by 20–25 different development teams with, in total, around 40–80 developers per team assigned to different projects. The requirements legacy database amounts to a very complex and large set of requirements, at various abstraction levels, in the order of magnitude of 20,000 entities. This makes it an example of the VLSRE (very-large scale RE) context [58]. Both the flow of new requirements (added to and removed from the scope of platform projects) and the scoping decisions associated with this flow may change frequently and rapidly. This exposes the project management to a series of unplanned, and often difficult, decisions where previous commitments have to be changed or cancelled.

3.1. Organisational set-up

Several organizational units within the company are involved in the development. For this case study, the relevant units are: the *requirements unit* that manages the scope and the requirements; the *software unit* that develops the software for the platform; and the *product unit* that develops products based on the platform releases. In addition, there is a *usability design unit* responsible for designing the user interface. Within each unit there are several groups of specialists divided by technical area. These specialists are responsible for the work in various stages of the development process. For this study, the most essential groups are the *requirements teams (RTs)* (part of the requirements unit) that, for a specific technical area, define the scope, and elicit and specify system requirements, and the *development teams (DTs)* (part of the software unit) that design, develop and maintain software for the (previously) defined requirements. 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 overall scope, which includes coordinating the RTs. In the DTs there are several roles, namely

- *Development team leader* who leads and plans the team's work for the implementation and maintenance phases;
- *Development 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* who monitor and coordinate the DTs and interact with the requirements architects. For the product development unit in this study, we focus on the *system testing* task from the viewpoint of the functionality and quality of the platform produced by the software unit.

3.2. Phase-based process

The company used a stage-gate model. There were *milestones* (MS) for controlling and monitoring the project progress. In particular, there were four milestones for the requirements management and design (MS1–MS4) and three milestones for the implementation and maintenance (MS5–MS7). For each of these milestones, the project scope was updated and baselined. The milestone criteria were as follows:

- MS1: At the beginning of each project, RT roadmap documents were 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, features contained a description sufficient for enabling estimation of its market value and implementation effort, both of which were obtained using a cost-value approach [37]. These values were the basis for initial scoping inclusion for each technical area when the features were reviewed, prioritized and approved. The initial scope was decided and baselined per RT, guided by a project directive and based on initial resource estimates given by the primary receiving (main) DT. The scope was then maintained in a document called feature list that was updated each week after a meeting of the change control board (CCB). The role of the CCB was to decide upon adding or removing features according to changes that occur.
- MS2: The features were refined to requirements and specified by the RTs, and assigned to their main DTs, responsible for designing and implementing the feature. The requirements were reviewed with the main DTs and were then approved. Other (secondary) DTs that were also affected by the features were identified. The DTs make an effort estimate per feature for both main and secondary DT.
- MS3: The DTs had refined the system requirements and started designing the system. The set of secondary DTs were refined along with the effort estimates, and the scope was updated and baselined.

Please cite this article in press as: E. Bjarnason et al., Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering, Inform. Softw. Technol. (2012), http://dx.doi.org/10.1016/j.infsof.2012.04.006

- MS4: The requirements refinement work and the system design were finished, and implementation plans were produced. The final scope was decided and agreed with the development resources, i.e. the software unit.
- MS5: All requirements had been developed and delivered to the platform.
- MS6: The software in the platform had been stabilized and prepared for customer testing.
- MS7: Customer-reported issues had been handled and the software updated. The software was ready to be released.

According to the company's process guidelines, the majority of the scoping work should have been done by MS2. The requirements were expressed using a domain-specific, natural language, and contained many special terms that required contextual knowledge to be understood. In the early phases, requirements contained a customer-oriented description while later being refined to detailed implementation requirements.

3.3. Agile development process

In order to meet the challenges of managing high requirements volatility, the case company was introducing a new development process at the time of this study. The size and complexity of the software development, including the usage of product lines, remained the same irrespective of the process used. The new process has been influenced by ideas and principles from the agile development processes eXtreme programming (XP) [7] and Scrum [67]. The phase-based process was replaced by a continuous development model with a toll-gate structure for the software releases of the software product line (to allow for coordination with hardware and product projects, see P1 below). The responsibility for requirements management was transferred from the (previous) requirements unit, partly into the business unit and partly into the software unit. The following agile RE practices were being introduced:

- *One continuous scope and release-planning flow* (P1). The scope for all software releases is continuously planned and managed via one priority-based list (comparable to a product backlog). The business unit gathers and prioritizes features from a business perspective. All new high-level requirements are continuously placed into this list and prioritized by the business unit. The software unit estimates the effort and potential delivery date for each feature based on priority and available software resource capacity. Development is planned and executed according to priority order. Planning and resource allocation is handled via one overall plan which contains all the resources of the software unit. The scope of the platform releases are synchronized with the product releases by gradual commitment to different parts of the scope. Critical scope is requested to be committed for specific product releases, while non-critical features are assigned to product releases when they are implemented and ready to be integrated into the platform.
- *Cross-functional development teams* (P2) that include a customer representative assigned by the business unit (comparable to the agile practice of customer proxy) have the full responsibility for defining detailed requirements, implementing and testing a feature (from the common priority-based list). Each new feature is developed by one cross-functional team specifically composed for that feature. The different software engineering disciplines and phases (e.g. requirements, design and test) are performed in an integrated fashion and within the same team. The team has the mandate to decide on changes within the value, effort and time frames assigned for the feature.
- *Gradual and iterative detailing of requirements* (P3). The requirements are first defined at the high level (as features in the priority-based list) and then iteratively refined in the development teams into more detailed requirements as the design and implementation work progresses.
- *Integrated requirements engineering* (P4). The requirements engineering tasks are integrated with the other development activities. The requirements are detailed, agreed and documented during design and development within the same cross-functional development team through close interaction between the customer representative and other team members, e.g. designers, developers and testers.
- *User stories and acceptance criteria* (P5) are used to formally document the requirements agreed for development. User stories define user goals and the acceptance criteria define detailed requirements to fulfill for a user story to be accepted as fully implemented. The acceptance criteria are to be covered by test cases.

This study mainly focuses on the situation prior to introducing the new agile way of working, i.e. for projects working as described in Section 3.2. The agile RE practices covered in this paper were defined in the company's internal development process at the time of the study. Practices P1 and P2 were being used in the projects, while P3 was partly implemented, and P4 and P5 were in the process of being implemented. Thus, it was not possible to investigate the full impact of the new agile RE practices at the time of this study. Nevertheless, the study investigates how these (new) practices are believed to affect the overscoping situation, i.e. which causes and root causes may be impacted by the agile RE practices and, thus, lead to reducing overscoping and its effects.

4. Research method

The study was initiated due to a larger transition taking place within the case company and with the aim of understanding the differences between the scoping processes of the phase-based process and the new agile development process. Our previous research into scoping [71] served as the basis for identifying research questions aimed at seeking a deeper understanding of overscoping as a phenomenon. In order to obtain detailed insight, an explanatory approach [60] was taken and the study design was based on the specific company context and the authors' pre-understanding. (These investigations can then be broadened in future studies.) Existing knowledge from literature was taken into account in interpretation and validation of the results.

A single-company explanatory case study [60] was performed using mainly a qualitative research approach complemented by a quantitative method for some of the data gathering. Qualitative research is suitable for investigating a complex phenomenon (such as overscoping) in a real-life context where it exists [48] (such as our case company). In this study, practitioners' perceptions of overscoping were studied through interviews where the verbalized thoughts of individuals with a range of different roles at the case company were captured [48,60]. An overview of the research method is shown in Fig. 1.

The case study was performed in three phases, see Fig. 1. In the first phase, the industrial experience of one of the authors was used to formulate a hypothesis concerning possible (assumed) causes of overscoping and (assumed) effects which may result from overscoping. This hypothesis was used as a starting point in creating the interview instrument [69] for the interviews, which took place in the second phase of the study. In the third phase, the interview results were presented to (another) six practitioners from the same company and validated by using a questionnaire (see Section 6 for more details and [69] for the validation questionnaire). This was done to reduce the risk of inappropriate (false) certainty of the correctness of the results [60].

Please cite this article in press as: E. Bjarnason et al., Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering, Inform. Softw. Technol. (2012), http://dx.doi.org/10.1016/j.infsof.2012.04.006

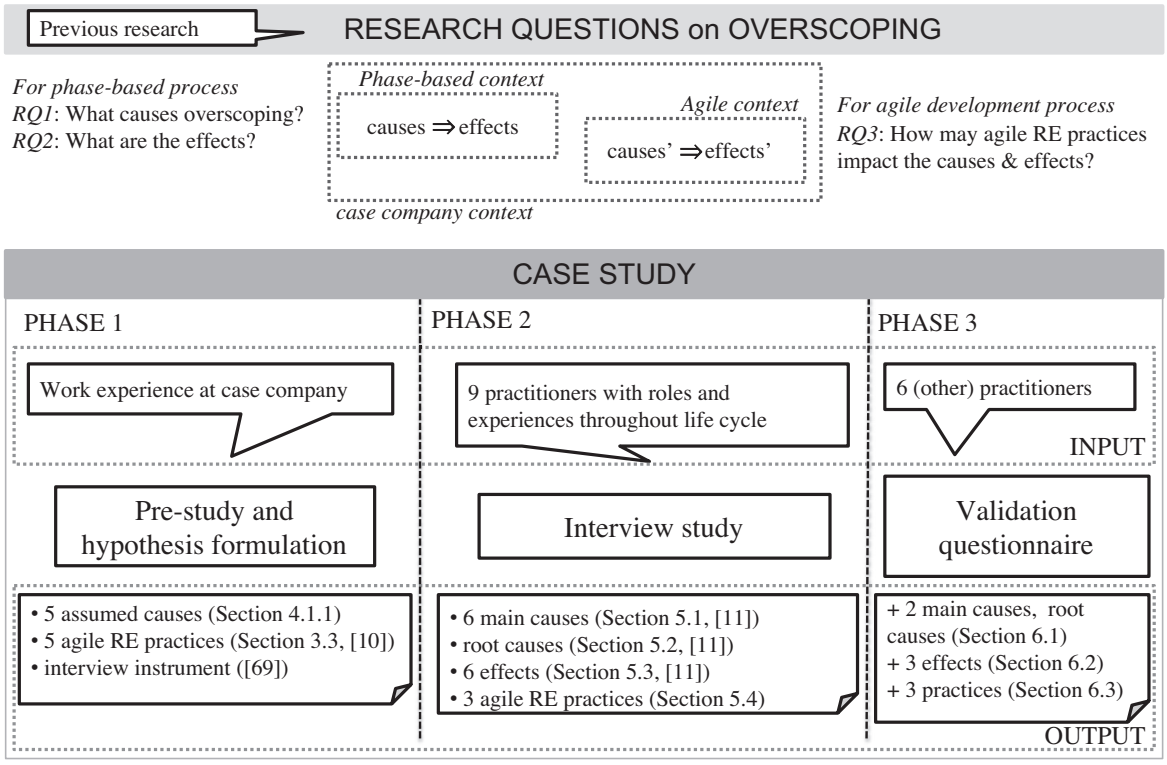

Fig. 1. Overview of research method for case study.

4.1. Phase one: pre-study and hypothesis generation

The purpose of the first phase of the study was to formulate a hypothesis on overscoping and prepare for the interviews. The experience of one of the authors (who has worked at the case company, with experience in several areas including coding, design, requirements engineering and process development) was used to identify possible (assumed) causes and effect of overscoping. In addition to these assumptions for the phase-based way of working, this author also identified the agile RE practices being introduced at the case company. These practices were assumed to impact one or more of the issues believed to cause overscoping in the phase-based process. If these assumptions were correct, applying the new practices should then result in reducing (or eliminating) the effects connected to those causes, and thus reduce (or eliminate) overscoping. In order to avoid selecting a set of assumptions biased by only one person, a series of brainstorming sessions around the hypothesis were conducted within the group of researchers involved in this study (i.e. the authors). The resulting (updated) hypothesis was then used as the main input in creating an interview study instrument (accessible online [69]).

4.1.1. Formulated hypothesis

The hypothesis formulated for this study is that overscoping is caused by a number of factors, and that by addressing one or more of these factors, e.g. through agile RE practices, the phenomenon of overscoping may be alleviated, or even eliminated. The following five factors were identified as assumed causes for overscoping in phase one:

- *continuous requirements inflow via multiple channels* (C1) was assumed to cause overscoping by the many inflows increasing the difficulty of defining a realistic scope for multiple parallel projects. Requirements continuously arrive from the market, as well as, from internal stakeholders. This inflow was managed by batching those requests into one or two projects per year. It was a challenge to manage the execution of multiple parallel projects, while handling requests for new features and requirements, as well as, requests for changes to the agreed project scope.
- *no overview of software resource availability* (C2) was assumed to cause overscoping due to the challenge of balancing the size of the total scope for several (parallel) development projects against the (same) set of development resources. The resource allocation for the software development unit was handled at the DT level, i.e. there was no total overview of the load and available capacity of all the development resources of the software unit.
- *low DT involvement in early phases* (C3) was assumed to contribute to defining unrealistic and unclear requirements in the early phases, that are later deemed too costly or even impossible to implement, thus causing overscoping. The development teams were not very involved in the early project phases (MS1–MS4) with providing cost estimates and feedback during requirements writing.
- *requirements not agreed with DT* (C4) was assumed to cause overscoping due to not ensuring that the suggested scope was feasible and understandable. The requirements specification was not always agreed with the development teams at the handover point (MS2). Even if there was a formal review by DTs, we assumed that there was a low level of commitment from DTs. Furthermore, this low level of agreement was assumed to lead to low DT motivation to fulfil the requirements defined by the RTs.
- *detailed requirements specification is produced upfront* (C5) by the requirements teams by MS2 before the design starts was assumed to cause overscoping by limiting the room for negotiating requirements that could enable a more efficient design and realistic development plan. Furthermore, time and cost overhead for managing such changes was also assumed to contribute to overscoping.

Please cite this article in press as: E. Bjarnason et al., Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering, Inform. Softw. Technol. (2012), http://dx.doi.org/10.1016/j.infsof.2012.04.006

4.2. Phase two: an interview study at the case company

In phase two, semi-structured interviews with a high degree of open discussion between the interviewer and the interviewee were held. The hypothesis provided a framework that helped to discuss, explore and enrich the understanding of this complex phenomenon. To avoid imposing this hypothesis on the interviewees, the discussion both on overscoping in general and on the detailed causes and effect was always started with an open ended question. In addition, the interviewees were asked to talk freely about the roles and phases she had experience from at the beginning of the interviews. In order to separate between the situation with the phase-based process and with the new agile RE practices, the impact of the new practices was discussed specifically in a (separate) section at the end of the interviews.

Our aim was to cover the whole process from requirements definition through development (design, implementation and testing) to the resulting end product (quality assurance, product projects), mainly for the phase-based process. This was achieved by selecting people with experience from all the relevant organizational units (see Section 3) to be interviewed and thereby catch a range of different perspectives on the phenomenon of overscoping. Nine people in total were selected to be interviewed. Two of the interviewees with identical roles requested to have their interviews together. The roles, organizational belongings, and relevant experience of the interviewed persons within the case company for the phase-based process can be found in Table 1. 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 unit with a letter S.

The interviews were scheduled for 90 min each with the possibility to reduce time or prolong it. All interviews were recorded and transcribed, and the transcripts sent back to the interviewees for validation. For each interview, the transcript was 7–10 pages long and contained in average 3900 words. Transcription speed varied from 3 to 7 times of recorded interview time. The coding and analysis was done in MS Excel. The underlying section structure of interview instrument, i.e. causes, effects and agile RE practices, were numbered and used to categorize the views of the interviewees. For each interview, the transcribed chunks of text were placed within the relevant sections and, if so needed, copied to multiple sections. Relationships between different categories, as well as, the level of agreement on causes, effects and agile RE practices were noted in specific columns. The viewpoints of the two practitioners interviewed together (interviewees Ra and Rb) were separated into different columns in order to allow reporting their individual responses.

Table 1
Interviewees roles (for phase-based process), organizational belonging and length of experience for each role within the company (see Section 3.1).

Code	Organizational unit	Role (s) within company	Years within role
Ra	Requirements	RT leader	5
Rb	Requirements	RT leader	2
Rc	Requirements	Requirements architect	3
Pd	Product	System test manager	7
Se	Software	Tester	3
Sf	Software	Software project manager	2
		DT leader	2
		Developer	2
Sg	Software	Quality manager	3
Sh	Software	DT requirements coordinator	1
		Developer	2
		DT leader	1
Si	Software	DT requirements coordinator	7

4.3. Phase three: validation of results via questionnaire

To further strengthen the validity of the results from the interviews a set of six (additional) practitioners at the case company was selected in phase three, see Table 2. To ensure that these six practitioners understood the results correctly and in a uniform way, the interview results were presented to them. During the meeting the participants could ask for clarifications and comment on the results, especially when they disagreed or had other different or additional viewpoints. In order to gather their views on the results in a uniform and non-public way, the participants were asked to fill out a questionnaire (available online at [69]) stating to which degree they agreed to the results and if additional relevant items had been missed. Due to limited availability of the participants a total of three such sessions were held. Each session was scheduled for 90 min with the possibility to extend or decrease the time as needed. The results from the questionnaire can be found in Section6.

5. Interview results

The causes and effects of overscoping derived from the interviews performed in phase two of the study (see Fig. 1) are outlined in Fig. 2 and described in the following sections. Section 5.1 covers the main causes for overscoping, while the root causes are reported in Section 5.2 and the effects in Section 5.3. The findings from the interviews concerning how the agile RE practices may address overscoping are described in Section 5.4. The outcome of the validation questionnaire (phase 3) on these results is reported in Section 6.

5.1. Causes of overscoping (RQ1)

The viewpoint of each interviewee concerning the causes of overscoping was categorized and matched against the hypothesis regarding the assumed causes of overscoping (C1–C5, see Section 4.1.1). In addition, five of the eight interviewees were found to describe a sixth main cause for overscoping, namely C6 *unclear vision of overall goal*. A lack of (clearly communicated) strategy and overall goals and business directions for software development led to unclarities concerning the intended direction of both software roadmaps and product strategies, as well as, unclear business priority of project scope. The interviewees described how this cause (C6) led to scope being proposed primarily from a technology aspect, rather than from a business perspective, and without an (agreed) unified priority. Instead, most of the scope of a project was claimed to be critical and non-negotiable.

The interviewee results around the main causes of overscoping are shown in Table 3. The opinions of the interviewees have been classified in the following way:

- *Experienced*: the cause (both its occurrence and its impact on overscoping) 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 had observed it or 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 caused overscoping.
- *Not mentioned*: even though within expected experience for role.
- *NA*: not within the expected experience for the role (according to the process).

Please cite this article in press as: E. Bjarnason et al., Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering, Inform. Softw. Technol. (2012), http://dx.doi.org/10.1016/j.infsof.2012.04.006

Table 2
Questionnaire respondents: roles and organizational belonging (for phase-based process), and length of experience within company (see Section 3.1 for descriptions of organizational units and roles).

Organizational unit	Role(s)	Years within company
Software	Software project manager, DT leader	4
Software	Tester	7
Software	DT requirements coordinator, DT leader	5
Requirements	Requirements architect	5
Requirements	RT leader	13
Product	System test manager	15

All interviewees had *Experienced* or *Agreed* to overscoping as a challenge, and a majority had *Experienced* or *Agreed* to causes 1–3. No interviewees *Disagreed* to any of the causes, though causes 4 and 5 both had less than a majority of *Experienced* or *Agreed* interviewees. Causes 4–5 were *Not mentioned* by all, while cause 6, which was derived from 5 of the interviewees, was *Not mentioned* by the others.

The entries marked *NA* (Not Applicable) indicate that the interviewee in her role was not expected to have experience of that cause. The system test manager (Pd) and the quality assurance manager (Sg) were classified as NA for C2, C3 and C4 since they merely observed the requirements flow from their management-level positions and were not directly involved in the early phases of the projects. In addition, Sg was also classified as NA for C5 due to lack of contact with the SRS. Furthermore, the software tester (Se), who had no insight into project planning, was categorized as *NA* for the causes C1 and C2.

For all assumed causes there were some counts of *Partly agreed,* namely:

- *continuous requirements inflow via multiple channels* (C1). The quality manager (Sg) mentioned the continuous inflow of requirement changes after setting the scope as causing overscoping, but no root causes prior to this milestone, and is therefore classified as 'Partly agreed'.
- *no overview of software resource availability* (C2). One of the DT requirements coordinators (Si) is noted as 'Partly agreed' to this cause, due to believing that a better overview of available resources would not alleviate the overscoping to any greater extent. In contrast, another interviewee (Sf) saw this as a strong cause for overscoping; 'There was no control of what people were working with. There were different DT leaders who just sent out [tasks]'.
- *low DT involvement in early phases* (C3). Both DT requirements coordinators (Sh, Si) were categorized as 'Partly agree' since the involvement from the rest of the DT including producing cost estimates was low, even though they personally had experienced good cooperation with the RT leaders during MS1–MS2. This lack of involvement was seen by the DT tester (Se) as leading to an unrealistically large scope being specified, 'The full view of requirements would be improved by including input from more roles, and a more realistic scope could be identified earlier on.'
- *requirements not agreed with DT* (C4). The DT requirements coordinators (Sh, Si) believed that the requirements were understood and agreed with the DT at MS2, though the DT did not commit to implementing them at that point. One of them (Sh) mentioned that the system requirements specification was primarily agreed with the DT requirements coordinators and not with developers and testers in the DT.
- *detailed requirements specification produced upfront* (C5). One of the RT leaders (Rb) had an agile way of working and did not produce a detailed requirements specification upfront, but instead regularly and directly interacted with the DT. This increased insight into the DT enabled a more flexible discussion around scope and detailed requirements, led to overscoping being experienced as a more manageable challenge by Rb. The other RT leader (Ra, interviewed together with Rb) did not mention

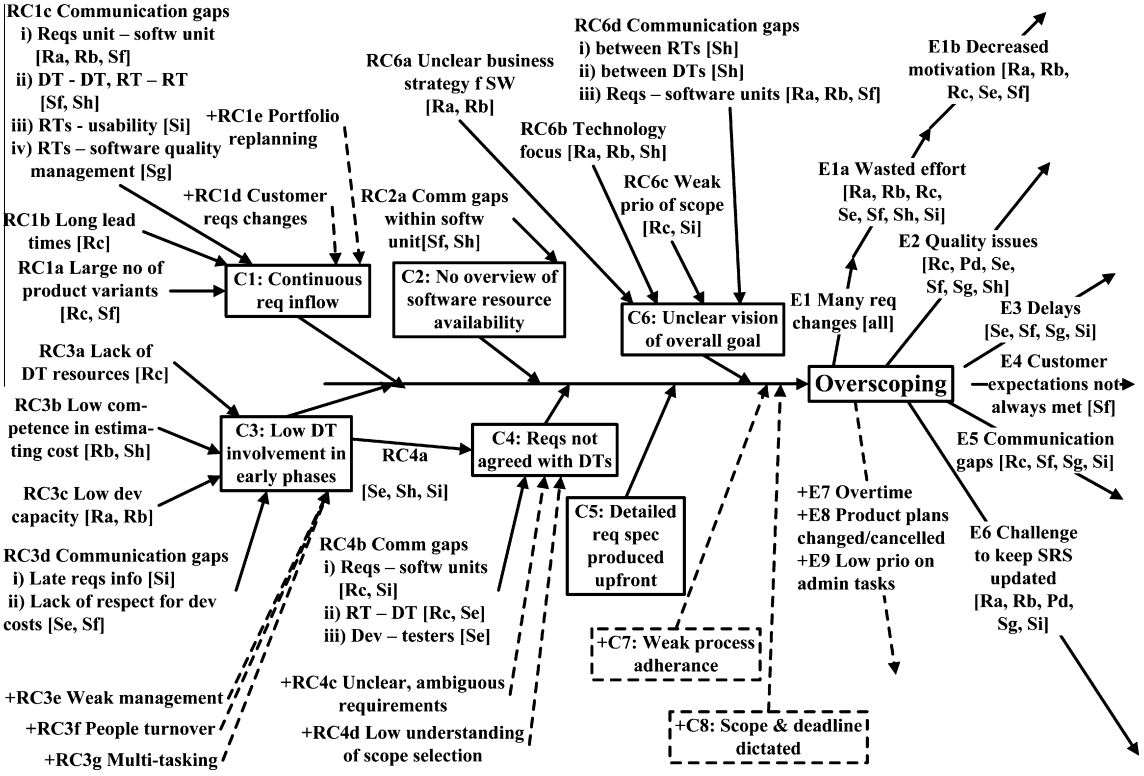

Fig. 2. Overview of all found causes (C), root causes (RC) and effects (E) of overscoping. Items derived from questionnaire noted with + and dashed lines. Interviewee code (see Section 4.2) noted within brackets.

Please cite this article in press as: E. Bjarnason et al., Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering, Inform. Softw. Technol. (2012), http://dx.doi.org/10.1016/j.infsof.2012.04.006

C5 as causing overscoping, but agreed to Rb's conclusions and was noted as 'Partly agreed'. Ra had the opposite experience, i.e. of producing and relying on a requirements specification, and then not staying in touch with the DT during the later phases of development (after MS2) who then developed software that was usually different from what was specified in the SRS. One of the DT interviewees (Sf) believed that the (wasted) effort of producing and agreeing to detailed requirements upfront (for features that were later descoped) increased the overscoping since it hindered those resources from working on viable features. Another interviewee (Sh) said: 'At this point [MS2] we [DT] approved a lot of things, because we liked what they [RT] wrote here and we really wanted that functionality then we [DT] started to over commit.'

5.2. Root cause analysis (RQ1)

To provide a deeper understanding the interviewees were asked to describe what may be triggering overscoping, i.e. the root causes of overscoping. These root causes have been grouped according to the main cause (C1–C6, outlined in Sections 4.1 and 5.1) that they affect. A full picture of the cause-effects relationships for overscoping identified through this study is depicted in Fig. 2. The results around root causes from both the interviews and from the questionnaire are also summarized in Table 4.

- *root causes of C1 (continuous requirements inflow via multiple channels)*. A number of requirement sources besides the regular requirement flow (defined by the RTs) were mentioned as contributing to the continuous inflow. These include: requirements produced by defining many different product variants in the product line (RC1a); and, many late new market requirements and changes incurred by the long project lead times (RC1b) compared to rate of requirements change. Furthermore, communication gaps (RC1c) were mentioned as causing additional requirements inflow through-out the project life cycle. These consist of communication gaps between the requirements unit and the software unit (RC1ci) which resulted in the software unit preferring to work according to their own software-internal roadmap containing a large amount of requirements not agreed with the requirements unit. Communication gaps between technical areas, both for RTs and for DTs, (RC1cii) led to indirect requirements between DTs being discovered after the initial scope selection at MS2, which greatly increased the amount of implementation required. The impact of these indirect requirements was especially large for DTs responsible for service-layer functionality like display frameworks and communication protocols. Furthermore, communication gaps between usability design and the RTs (RC1ciii) resulted in additional functional requirements appearing in usability design specification, sometimes in conflict with RT requirements. And, finally, due to lack of communication between the software quality managers and the requirements unit (RC1civ), requirements on quality aspects were not defined and prioritized together with the RT requirements, but managed separately in a later phase.
- *root causes of C2 (no overview of software resource availability)*. The lack of overview of available software development resources was believed to be a consequence of communication gaps within the software unit and between the DTs (RC2a). The organizational structures and the high scope pressure were seen to result in each DT focusing on their own areas rather than striving for cooperation and good communication with other DTs. One interviewee described that enabling DTs to coordinate their plans had the effect of improving the scoping situation by increasing the delivery rate and efficiency, 'We tried to solve the overscoping by enabling the DTs to co-plan and deliver incrementally. This resulted in more deliveries and increased efficiency.' (Sf)
- *root causes of C3 (Low DT involvement in early phases)*. Several interviewees described that software resources were rarely available in early project phases (RC3a) due to development and maintenance work for previous projects. Rc said: 'by MS2, but it was hard to get [DT] resources. That probably was the problem.' In addition, weak and incorrect cost estimations (RC3b) were mentioned as leading to including too much into the project scope. In contrast, low development capacity of the software unit (RC3c) caused by bad architecture was believed by the two RT leaders to be the main reason for overscoping. Furthermore, gaps in the communication (RC3d) between the requirements unit and the software unit were mentioned as causing low DT involvement. For example, interviewees mentioned that early DT involvement was often postponed due to a lack of understanding within the software unit for the importance of this work. However, the opposite was also mentioned, namely that the DTs received requirements information too late (RC3di) which then resulted in extending the project scope without realistic plans. Similarly, the cost estimates for both development and testing were not always respected (RC3dii). In contrast, close cooperation between the RTs and the DTs were experienced (by Rc) to lead to an early uncovering of problems, thereby enabling definition of more stable requirements that were then successfully implemented.
- *root causes of C4 (requirements not agreed with DTs)*. Low DT involvement in the early phases (C3, RC4a) was seen as leading to weak agreement and commitment to the requirements, by all three interviewees with experience from planning DT work (Se, Sh, Si). The interviewees connected the level of requirements agreement with the level of communication around requirements (RC4b), i.e. RTs and DTs that communicated well also tended to have a mutual understanding and agreement of the requirements. Due to variations in communication between teams, the view on C4 varied between interviewees (see Section 5.1). Even so, one interviewee (Sh) who had experienced good cooperation with the RT mentioned that the different organizational belongings (RC4bi) caused timing issues due to different priorities for different units. In addition, communication gaps between RTs and DTs (RC4bii) including no contact between testers and RT leaders were caused by physical and organizational distances and resulted in weak DT agreement on the requirements. Weak communication on requirements and design between developers and testers (RC4biii) was also mentioned (by Se) as causing weak requirements agreement.
- *root causes of C5 (detailed requirements specification produced upfront)*. The phase-based process defined that a requirements specification should be produced by MS2, therefore no further root causes have been identified for this cause.
- *root causes of C6 (unclear vision of overall goal)*. The RT leaders (Ra and Rb) described that the lack of clear business strategy (RC6a) and vision that could guide them in defining a roadmap resulted in proposing a project scope from a pure technology standpoint (RC6b). A weak and un-unified business priority (RC6c) of the proposed scope (almost everything was 'critical') was described (by Si) as pushing the DTs to commit to unrealistic project plans. In addition, Rc mentioned that the lack of unified priority hindered the project management from effectively addressing the overscoping. Furthermore, several communication gaps (RC6d) were seen to contribute to this cause. Weak communication both between RTs (RC6di) and between DTs (RC6dii) were described by Rc as resulting in weak scope coordination between functional areas, as well as, conflicts and lack of clarity concerning the overall goal. Finally, both RT

Table 3
For each identified main causes of overscoping, the number of interviewees per response category (see Section 5.1) and organizational unit (Reqs, etc, see Section 3.1).

	Overscoping as a challenge			C1 Continuous req inflow			C2 No overview of softw resources			C3 Low DT involvm in early phases			C4 Reqs not agreed with DTs			C5 Detailed reqs specification produced upfront			C6 Unclear vision of overall goal		
	Reqs	Softw	Product	Reqs	Softw	Product	Reqs	Softw	Product	Reqs	Softw	Product	Reqs	Softw	Product	Reqs	Softw	Product	Reqs	Softw	Product
Experienced	2	5	1	1	3	1	1	2		3	1		1	1			2		3	2	
Agreed	1			2			2				1			1		1	1				
Partly agreed					1			1			2			2		1					
Disagreed																					
Not mentioned													2			1	2			3	1
NA					1			2	1		1	1		1	1			1			

leaders described that communication gaps and low common understanding between the requirements unit and the software unit (RC6diii) of the overall goal resulted in the project scope being decided to a large extent by the DTs, and not (as the process stated) by the RTs.

5.3. Effects of overscoping (RQ2)

The interviews uncovered the following six main effects of overscoping (marked as E1 to E6, see Fig. 2):

- *many requirement changes after the project scope is set (E1)*. All interviewees had experienced that overscoping caused requirement changes to take place after the project scope was set (at MS2). As the projects proceeded and the overload was uncovered large amounts of features were removed from scope (descoped). The phenomena was so common that the phrases 'overscoping' and 'descoping' have become part of company vocabulary. This descoping of already started features was a waste (E1a) of both RT and DT effort and led to frustration and decreased motivation (E1b) to work with new requirements. As interviewee Sh 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.' However, the many requirement changes were experienced by Pd as having only minor impact on the system testing. They merely adjusted the test plans, and rarely wasted any effort due to this effect.
- *quality issues (E2)*. All interviewed practitioners involved after MS4 (when development started, Rc, Pd, Se, Sf, Sg, Sh) mentioned that software quality was negatively affected by overscoping both due to the high workload and due to the many requirement changes. The software quality manager Sg expressed, 'If you get too much scope, you get quality problems later on and you haven't got the energy to deal with them.' Similarly, 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.' Furthermore, the lack of respect for development costs (C3dii) in the earlier phases was mentioned by the software tester (Se) to contribute to insufficient testing and subsequent quality issues.
- *delays (E3)*. The overscoping and subsequent overloading of the DTs was described by several practitioners as resulting in delayed deliveries being the norm rather than the exception. In addition, overscoped DTs were often forced to commit to customer-critical requests and changes which in turn resulted in even more delays and quality issues (E2). One DT interviewee (Sf) stated that 'our team was always loaded to 100% at MS4, which was too much since there were always customer requests later on that we had to handle. That meant that we were forced to deliver functionality with lower quality or late.' The same situation was described by the quality manager (Sg) who said: 'Even if we decided on a scope for MS4, there were extremely many changes underway, so we were never ready by MS5, as we had said, but were delayed.'
- *customer expectations are not always met (E4)*. Overscoping was mentioned by a few interviewees as resulting in sometimes failing to meet customer expectations. For example, customers sometimes file change requests for features that had previously been removed due to overscoping. In addition, overscoping caused by requiring a large number of products (RC1a) with different display sizes and formats was experienced by interviewee Sf as resulting in releasing products with faulty software, e.g. misplaced icons.
- *communication gaps (E5)*. Overscoping and overloading an organization was described as leading to several communication gaps; between the requirements and software units; within the software unit itself, between DTs (Sg, Si) and between DTs and software project managers (Sf); and between the software and the product unit. For example, the many descoped features (E1) and wasted effort (E1a) resulted in distrust between the requirements unit and the software unit, so much so that the software unit defined their own internal roadmap without coordinating this with the requirements unit. Furthermore, invalid error reports filed by the system testers based on an unreliable SRS (caused by E1–E6) caused an increase both in work load and in frustration at the software unit and, consequently friction and widened communication gaps between these units.
- *challenge to keep the SRS updated (E6)*: The situation caused by overscoping, with a high workload and many late requirement changes (E1), increased the challenge of keeping the SRS updated. The practitioners mentioned that in an overscoping situation the task of updating the SRS was given low priority (partly caused by E1b). Furthermore, the amount of required updates both for changed and descoped requirements was increased (Ra, Rb, Pd, Sg, Si) by producing the requirements upfront (C5) with a low level of DT agreement (C4). The RT leaders (Ra, Rb) had also experienced that many requirement-related changes were made during development without informing the RTs (or the system testers), many of which might have been a result of insufficient DT involvement in the early phases (C3).

5.4. Impact of agile RE practices (RQ3)

The general opinion of the interviewees on the situation after introducing the agile RE practices (see Section 3.3) is that even though some overscoping is still experienced, it is a more manageable challenge than with the previous phase-based process. For example, there is less descoping and most of the features worked on by the software unit now continue until completion (Si). Interviewee Sg said: 'We still have overscoping in all projects. But, it is more controlled now and easier to remove things without having

Please cite this article in press as: E. Bjarnason et al., Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering, Inform. Softw. Technol. (2012), http://dx.doi.org/10.1016/j.infsof.2012.04.006

Table 4
Summary of all identified causes and root causes of overscoping: Number of responses for interviewees (see Section 5 for details) and for Questionnaire responses per level of agreement (see Section 6 for details). Additional items from questionnaire responses are marked with +.

	Mentioned as causes/root causes by # of interviewees (9 in total)	Number of questionnaire responses (6 in total)				
		Experienced	Agree	Partly agree	Disagree	Do not know
Overscoping (as a challenge)	9	6				
C1: Continuous reqs. inflow via multiple channels	8	4	2			
(a) Large number of product variants	2	3	1	2		
(b) Long lead times	1	4	2			
(c) Communication gaps	6	3	1	1		1
(i) Between reqs & software unit	3	3	2		1	
(ii) Between RT–RT and DT–DT	2	3	2			1
(iii) Between RT and usability design	1	2	1	1		2
(iv) Between RT and software quality managers	1		2			4
(+d) Customer requirements changes (many and late)		3				
(+e) Product portfolio re-planning		1				
C2: No overview of software resource availability	6	2	3		1	
(a) Communication gaps within software unit	2	1	2	1	1	1
C3: Low development team involvement in early phases	7	1	2	2	1	
(a) Lack of DT resources for pre-development work	1	2	2	2		
(b) Low competence in estimating cost	2	2	1	3		
(c) Low development capacity	2	1	1	2	1	1
(d) Communication gaps	3					
(i). Late requirements information to DT	1	2		2	1	1
(ii). Lack of respect/understanding of development costs	2	2	3		1	
(+e) Weak leadership including ineffective communication		1				
(+f) Change of people during the project		1				
(+g) Multi-tasking		1				
C4: Requirements not agreed with development teams	5	2	2	2		
(a) Low DT involvement in early phases (C3)	3	2	2	1	1	
(b) Communication gaps	3	1	2	2	1	
(i) Between requirements and software units	2	1	2		1	2
(ii) Between RT and DT	2	1	1	2	1	1
(iii) Between developers and testers	1	1	1	1	2	1
(+c) Unclear and ambiguous requirements		3				
(+d) Low understanding of why particular scope is selected		1				
C5: Detailed reqs specification produced upfront	5	1	3	1	1	
C6: Unclear vision of overall goal	5	4	1		1	
(a) Unclear business strategy for software development	2	3	2	1		
(b) Technology focus when scope set	3	3	2		1	
(c) Weak priority of scope	2	3	2	1		
(d) Communication gaps		2	3			1
(i) Between RTs	1	1	3			2
(ii) Between DTs	1	2	2			2
(iii) Between requirements and software units	3	1	3		1	1
+C7 Weak process adherence		1				
+C8 Overall scope and deadline dictated from management		1				

done too much work.' Many of the interviewees stated that in theory the agile RE practices address overscoping, but that these practices also incur a number of new challenges. The following practices were mentioned by the interviewees as impacting some of the causes and/or root causes of overscoping:

- *one continuous scope & release-planning flow (P1).* This practice (which was implemented at the time of the interviews) was seen to address the root cause weak prioritization of scope (RC6c, mentioned by Rc, Pd, Sg, Sh) and the causes continuous requirements inflow via multiple channels (C1, mentioned by Se, Sf) and no overview of software resource availability (C2, mentioned by Sf, Sg), by enforcing that all scope and development resources are managed through a uniformly prioritised list.
- *cross-functional development teams (P2).* This practice (which was implemented at the time of the interviews) was seen to address several communication gaps, and, thus, impact causes C1–C4 by closing the gaps (identified as root causes) between RTs and DTs and between different functional areas. This practice was also believed to impact C5 (detailed requirements specification produced upfront) since detailing of requirements is now handled within the development teams together with the customer representative. Interviewee Sf said: 'It is an advantage that they [the team] sit together and can work undisturbed, and there is no us-and-them, but it is us. And when they work with requirements the whole group is involved and handshakes them.'
- *gradual and iterative detailing of requirements (P3).* This practice (which was partly implemented at the time of the interviews) was mentioned as directly impacting the cause C5 (detailed SRS produced upfront). Furthermore, this practice was also seen by Sf and Sg to reduce both the lead time for each high-level requirement (RC1b) and the amount of changes after project scope is set (E1) and, thus also reduce the amount of wasted effort (E1a, also mentioned by Ra, Rb).

6. Validation questionnaire on interview results

Overscoping was further investigated through the validation questionnaires [69], see Table 4. Each of the six respondents noted her level of agreement by using the following notation:

Experienced: I have experienced this (item and connection) to be valid.
Agree: I agree to this, but have not experienced it personally.
Partly agree: I agree to part, but not all, of this.
Disagree: I do not agree.
Do not know: I have no knowledge of this item or its impact.

6.1. Causes and root causes (RQ1)

A majority of the questionnaire respondents confirmed (i.e. *Experienced* or *Agreed* to) all main causes as contributing to overscoping, except C3 (low DT involvement) for which there was also one *Disagree* response. Causes C2, C3, C5 and C6 each had one count of *Disagree* from respondents with experience from the requirements unit. Two additional main causes were given by two respondents, namely *weak processes adherence* (+C7) and *dictation of scope and deadlines from management* (+C9). Furthermore, some additional root causes were given for C1, C3 and C4. For C3, two alternative root causes were given, namely turn-over of DT members as the project progressed (RC3f) and assigning the same resources to multiple parallel projects (RC3g). For C4 (*requirements not agreed with DT*) three respondents stated that this was caused by unclear and ambiguous requirements (RC4c), while one respondents suggested that DTs often lacked insight into why certain features and requirements were important, which is related to C6 (*unclear vision of overall goal*). All responses from the validation questionnaire on causes and root causes can be found in Table 4.

The impact of each main cause on overscoping was gauged by asking the questionnaire respondents to distribute 100 points over all causes according to the extent of their impact (see Table 5) C1 got the highest score in total and all six respondents, thereby indicating that the continuous requirements inflow was a main cause of overscoping. The second highest total score was given to C6 (*unclear vision of overall goal*), which all the participants from the software unit graded with 30–60, while the other participants graded this with 0 or 30. Causes C4, C5, +C6 and +C7 were seen as having a minor or no impact on the overscoping situation.

6.2. Effect of overscoping (RQ2)

In large, the questionnaire respondents had experienced or agreed to all the effects of overscoping identified from the interviews. The respondent from the product unit had no view on E5 or E6, while the requirements architect partly agreed E5. In addition, the respondents mentioned the following effects of overscoping: overtime (+E7); changed and sometimes cancelled product plans (+E8); low prioritization of administrative tasks (+E9). The full questionnaire response on effects is shown in Table 6.

In addition to stating the level of agreement to the identified effects of overscoping, the respondents were asked to grade their impact. The following notation was used:

Critical: Company or customer level.
Major: Project or unit level.
Medium: Team level.
Minor: Individual level.
None: No impact.

All the effects identified from the interviews were seen as having an impact. All effects except E5 (communication gaps) were seen as having major or critical impact by a majority of the participants. There were two counts of minor impact: one for E6 (keeping SRS updated) and one for +E7 (overtime).

Table 5
The total number of points reflecting the impact of each cause on overscoping. Each questionnaire respondent distributed 100 points. Effects of overscoping (RQ2). The columns show the number of points per responder (organisational belonging given in header, see Section 3.1).

	Total impact	Softw	Softw	Softw	Reqs	Reqs	Product
C1: Continuous reqs inflow via multiple channels	275	20	20	15	50	100	70
C2: No overview of software resource availability	60	10	20		20		10
C3: Low DT involvement in early phases	80		10	50	20		
C4: Requirements not agreed with DTs	10	5	5				
C5: Detailed reqs specification produced upfront	15	5	5	5			
C6: Unclear vision of overall goal	140	60	40	30	10		
+C7: Weak process adherence	0						
+C8: Overall scope and deadline dictated from top	20						20

Please cite this article in press as: E. Bjarnason et al., Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering, Inform. Softw. Technol. (2012), http://dx.doi.org/10.1016/j.infsof.2012.04.006

Table 6
Number of questionnaire responses on the effects of overscoping per level of agreement (notation described in Section 6) and per impact category (notation described in Section 6.2). Additional items derived from questionnaire marked with +.

	Mentioned by # of interviewees (9 in total)	Questionnaire responses (6 in total)									
		Agreement					Impact				
		Experienced	Agree	Partly agree	Disagree	Do not know	Critical	Major	Medium	Minor	None
E1: Many req changes after scope is set	9	5	1				4	2			
(a) Wasted effort	7	5	1				3	3			
(b) Decreased motivation	5	4	2				3	2	1		
E2: Quality issues	6	6					5	1			
E3: Delays	4	6					5	1			
E4: Customer expectations not always met	1	4	2				5	1			
E5: Communication gaps	4	2	1	1		1	2	1	3		
E6: Keep SRS updated	5	1	4			1		5		1	
+E7: Overtime		3					1		1	1	
+E8: Changed & cancelled product plans		1					1				
+E9: Administrative tasks not always performed		1					1				

Table 7
Number of questionnaire responses on the impact of agile RE practices on overscoping per level of agreement (notation described in Section 6). Additional practices identified through questionnaire responses are marked with +.

	Experienced	Agree (in theory)	Partly agree	Disagree	Do not know
P1: One continuous scope and release-planning flow	2	4			
P2: Cross-functional development teams	3	2			1
P3: Gradual and iterative detailing of requirements	2	2	2		
+P4: Company vision	1				
+P5: Open source development	1				
+P6: Incremental deliveries	1				

Table 8
Number of questionnaire responses per agreement category (described in Section 6) on the current situation at the case company with agile RE practices, as compared to when using phase-based process.

	Experienced	Agree	Partly agree	Disagree	Do not know
Overscoping is still a challenge	3	1	2		
There is less overscoping now	1	1	3	1	
Overscoping is more manageable now	1	3	1		1

6.3. Impact of agile RE practices (RQ3)

The questionnaire respondents mostly agreed to the three identified agile RE practices as impacting the challenge of overscoping, either through their own experience or by believing the practice should work in theory. Furthermore, some additional practices were mentioned as impacting overscoping: (+P4) clearer company vision (i.e. directly addressing C6), (+P5) open source development (limiting C1 by restricting what the customer can reasonably expect when large parts of the software are outside of company control) and (+P6) incremental deliveries (shorter cycles facilitate scope size control for each cycle). Table 7 contains the questionnaire responses on the impact of the agile RE practices on overscoping.

Finally, the respondents had all experienced, agreed or partly agreed that overscoping was still a challenge for the case company. The new agile process and practices are seen to, at least partly, address the situation and provided ways to better manage and control the extent of overscoping and its effects. The practitioners' responses concerning the current situation are shown in Table 8.

7. Interpretation and discussion

The results of this study corroborate that overscoping is a complex and serious risk for software project management [13,20,43] both for phase-based and for agile development processes. In addition, the results show that communication issues have a major impact on overscoping. This complements the work by Sangwan et al. and Konrad et al. [41] who mentioned that weak communication can cause project failures in large-scale development and global software engineering [63]. Moreover, our results extend the lists of effects of weak coordination proposed by Sangwan et al. [63] (long delays, leave teams idle and cause quality issues) by adding overscoping. Further research is needed to fully identify and address the factors involved. The results are discussed and related to other research in further detail, per research question, in Sections 7.1 (RQ1), 7.2 (RQ2) and 7.3 (RQ3). Finally, the limitations of this study and threats to validity of the results are discussed in Section 7.4.

7.1. Causes of overscoping (RQ1)

Our results indicate that overscoping is caused by a number of causes and root causes. These causes mainly originate from the nature of the MDRE context in which the company operates, but are also due to issues concerning organizational culture and structures, and communication. This was further highlighted by interviewees describing the additional cause C6 (unclear vision of overall goal) and two questionnaire respondents mentioning additional causes connected to lack of respect for the decision- and

Please cite this article in press as: E. Bjarnason et al., Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering, Inform. Softw. Technol. (2012), http://dx.doi.org/10.1016/j.infsof.2012.04.006

development process, i.e. C7 and C8. In contrast, practitioners with experience of good cooperation and well-communicating teams described overscoping as a less serious and more manageable challenge. This may explain all the *Disagree* questionnaire responses but one (i.e. C5).

We interpret the results around the six causes of overscoping identified through the interviews (see Section 5.1 and Fig. 2) as follows:

- *continuous requirements inflow from multiple channels (C1)*. We interpret the homogeneity of the interview and questionnaire results (see Tables 3 and 5) to mean that a large and uncontrollable inflow of requirements has the potential to cause overscoping when not managed and balanced against the amount of available capacity. This cause was also been identified by Regnell and Brinkkemper [59] and Karlsson at el. [38] as one of the challenges of MDRE. In addition to corroborating this challenge, our work also identifies that this continuous inflow of requirements can cause overscoping. The importance and seriousness of this factor are indicated by this cause scoring the highest total impact factor in the questionnaire (see Table 5). The extent to which this cause affects companies that operate in the bespoke requirements engineering context [59] requires further research.

  Our study also reveals that the inflow of requirements can be further increased by scope creep at the software management level through a software-internal roadmap (RC1ci, see Section 5.2). In effect, this hindered resources from being available for managing new customer requirements. Similar results have been reported by Konrad et al. who found that scope creep can result in problems with meeting customer expectations [41], i.e. effect E4 (see Section 5.3). Konrad et al. [41] propose addressing scope creep by increased understanding and traceability of customer requirements, and by creating an effective hierarchical CCB structure. The impact of these methods on overscoping remains to be evaluated.

- *no overview of software resource availability (C2)*. The majority of our responders (six of nine interviewees and five of six questionnaire respondents) had experienced or agreed to the lack of overview of available resources being a cause of overscoping. However, the questionnaire results suggest that the impact of this cause is not as critical as cause C1. This result is surprising, when considering the importance of management of the daily workload including coordination of tasks and activities reported by, e.g. Philips et al. [52]. The contrasting opinions of low development capacity (RC3c, held by RT leaders) and low respect for development costs (RCdii, held by DT roles) is interesting. This difference can be interpreted as a low understanding of each other's viewpoint around cost and an indication that this viewpoint is dependent on role (related to Jorgensen and Shepperd [33]). If the development capacity really is low is a different issue. Finally, this cause specifically includes the lack of overview, or awareness of the total load on the resources. To the best of our knowledge, this issue has not been empirically investigated. Rather software cost estimation research [33] mainly focuses on effort estimation and on optimizing resource assignment [45].
- *low development team involvement in early phases (C3)*. The results indicate that low development involvement in the requirements phase can cause overscoping (mentioned by 6 out of 9 interviewees and 5 out of 6 questionnaire respondents did not disagree to this). This confirms previous work that points out the need of early development involvement in requirements engineering, e.g. required by interdependencies between product management and software development [51]. Glinz et al. also mentioned that lack of communication between project management and development at requirements hand-off may lead to unsatisfactory results [27]. Similarly, Karlsson et al. reported that communication gaps between marketing [38] (requirements unit for our case company) and development, can result in insufficient effort estimates (i.e. RC3b) and in committing to unrealistically large features without considering the technical and scheduling implications [38]. Our results corroborate these results in that low involvement and weak communication in early phases may lead to problems later on, including overscoping. These communication issues may also exacerbate the problem of getting accurate and reliable effort estimates (RC3b). Furthermore, the fact that one questionnaire respondent expressed experiencing good communication and cooperation between requirements and development teams may also explain the one Disagree response for this cause. On the other hand, a surprising result from the validation questionnaire is that this cause (C3) was seen to influence overscoping less than cause C6 (unclear vision of overall goal) both in total (among all respondents) and by 2 of the 3 software respondents. These results indicate that there may be additional (uncovered) factors that influence the impact this cause has on overscoping.

  Finally, several methods have been proposed for addressing cause C3, e.g. negotiation of implementation proposals [25], model connectors for transforming requirements to architecture [47], cooperative requirements capturing [46] and involving customers in the requirements management process [35]. Goal-oriented reasoning can also provide constructive guidelines for architects in their design tasks [42]. If and to which degree the mentioned methods can alleviate overscoping by impacting this cause remains a topic for further research.

- *requirements not agreed with development team (C4)*. The results provide empirical evidence that weak agreement on requirements between requirements and software units can cause overscoping (all 6 questionnaire responders agreed to cause C4 and five interviewees mentioned C4 as a cause of overscoping). A significant root cause for this cause was found to be communication gaps, mainly between the requirements-related roles and the development and testing roles. This confirms the viewpoint of Hall et al. [29] that most requirement problems are actually organizational issues. In addition, this confirms the importance of seamless integration of different processes in collaborative work [22]. The impact of insufficient communication on software engineering has been reported as a general issue within requirements engineering and product management [12,25,29,35,38]. Surprisingly, C4 scored the lowest impact among all the causes and only two questionnaire responders (both from the software unit) rated this cause as having any (low) impact factor on overscoping. In contrast, cause C6 (weak vision of overall goal) was rated as having the largest impact on overscoping.
- *detailed requirements specification produced upfront (C5)*. Our results indicate that too much detailed documentation produced upfront may cause overscoping (mentioned by five interviewees and experienced, agreed or partly agreed to by five questionnaire respondents, see section 5.1). This complements other studies into documentation in software engineering projects. For example, Emam and Madhavji mentioned that in organizations which require more control the pressure to produce much detail is also greater [23]. Lethbridge reported that, for software engineers, there is often too much documentation for software systems, frequently poorly written and out of date [44]. Furthermore, Sawyer et al. [65] mention that premature

Please cite this article in press as: E. Bjarnason et al., Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering, Inform. Softw. Technol. (2012), http://dx.doi.org/10.1016/j.infsof.2012.04.006

freezing of requirements may cause scope creep and communication problems (both of which are identified as root causes of overscoping in our study) and suggest evolutionary prototyping as a remedy. Other remedies suggested for addressing excessive documentation include reuse of requirements specifications [24], as well as, simply creating less documentation [3]. The effectiveness of these methods for the risk of overscoping remains to be investigated. The differing views on this cause between respondents may be explained by their roles and relationship to RE. All the disagreeing questionnaire respondents for this cause worked with requirements related roles. These roles are more likely to consider detailed requirements specifications as positive and good, rather than an issue. However, these roles have less insight into the later phases when development takes place and the effects of overscoping are experienced. Three of the respondents with experience from later development phases had experienced C5 as causing overscoping. Furthermore, Berry et al. mentioned that when time for elicitation is short, i.e. there is a lack of upfront documentation (or lack of C5), the requirements usually end up as an enhancement or become descoped since all of the client's requests cannot be delivered [9]. Considering this, we conclude that both under specifying (as in [9]) and over specifying (as in our study) can cause overscoping and later descoping, and that it remains to be investigated how to strike a good balance.

- *unclear vision of overall goal (C6)*. Our study identifies that a lack of clearly communicated goals and strategy for software development may cause defining the project scope primarily from a technology perspective, rather than with a business focus, thereby contributing to overscoping. Overall this cause was graded as having the second largest impact on overscoping, despite one questionnaire respondent (an RT leader) disagreeing to this cause. Our results support the findings from related papers [4,18,20,40,49,61] that stress the importance of selecting requirements aligned with the overall business goals and discarding others as early as possible. In addition, failure of stakeholders to concur on project goals was found by DeMarco and Lister to pose the biggest risk for a project [20]. A method for early requirements triage based on management strategies was proposed by Khurum et al. [40]. Aurum and Wohlin have proposed a framework for aligning requirements with business objectives [4]. Rosca et al. mention that the most demanding characteristic of business is the likelihood of change which cannot be fully controlled [61]. This can be managed when business objectives are clear to the software developers, thus enabling them to manage a system requiring modifications while meeting the business objectives [18]. Finally, Karlsson et al. [38] mentioned the lack of common goals and visions as a challenge in achieving good cooperation, quoting their responders: 'If everyone has the same goal and vision, then everyone works in the right direction.'
- *weak process adherence (+C7) and scope and deadline dictated by management (+C8)*. These two causes were mentioned in the questionnaires, though none of them were seen as having any major impact on overscoping. Karlsson et al. [38] found that weak process adherence may be caused both by high process complexity, as well as, lack of time for process implementation. The latter could be a consequence of overscoping. The direction of causal relationship between overscoping and process adherence remains to be investigated.

7.2. The effects of overscoping (RQ2)

The results indicate that overscoping may lead to a number of effects (or consequences), many of which are judged to be serious and potentially very costly for the company. Several of the identified effects may be in line with held beliefs about what overloading a project with too much work may lead to. The aim of this study is to investigate if such beliefs can be supported by empirical evidence or not, and if more surprising phenomena arise in relation to a specific, real-world overscoping situation.

- *many changes after the project scope is set (E1)*. The results show that overscoping leads to a large number of scope changes (experienced by all responders and impact graded as critical or major by all six questionnaire responders). This confirms evidence provided by Harker and Eason [30] that requirements are not static and, thus, are hard to capture or classify. In addition, requirements volatility is mentioned as one of the challenges in MDRE by Karlsson et al. [38] and identified by Ramesh et al. as one of the 14 assumptions underlying agile software development [57]. Furthermore, origins of requirements volatility have been listed [30]. Despite this awareness, causes for requirements volatility have not been empirically explored. Our results highlight overscoping as one possible cause of late requirement changes. Furthermore, our results confirm that it is challenging to manage requirement changes.
- *quality issues (E2)*. The results indicate this as an important effect of overscoping (experienced and agreed for both interviews and questionnaires, and graded as having critical or major impact). This confirms that the quality of requirements engineering determines the software quality, as reported, e.g. by Aurum and Wohlin [6]. In addition, our results highlight overscoping as a potential reason for quality issues.
- *delays (E3)*. This study shows (with a high degree of alignment between interviewees and questionnaire responses) that delays can be an effect of overscoping. Within MDRE, delays in launching products can be very costly and result in loss of market shares [38,59,64,65]. Therefore, the insight that overscoping may have this effect is important evidence that indicates that overscoping is a (potentially) serious risk.
- *customer expectations are not always met (E4)*. Our results indicate that overscoping can have the effect of failing to meet customer expectations. This could be explained by an overloaded project having no time or capacity neither to analyse or implement new requirements, nor to validate if market or customer needs could have changed. Furthermore, Karlsson et al. reported failure to meet customer needs as one of the risks of developing products based on a technology focus (root cause RC6b) [38]. Another crucial part of producing software products that will satisfy the customers, as pointed out by Aurum and Wohlin, is working with RE throughout the project life cycle (as opposed to upfront requirements detailing, C5) [6]. The results of this study highlight the importance of selecting a feasible scope as one factor to consider when attempting to better understand and capture the customers' needs.
- *communication gaps (E5)*. Our results indicate that overscoping may cause increased communication gaps. (Roughly half of our interviewees and questionnaire respondents mentioned and agreed to this effect.) This may be explained by the tendency to deflect by blaming others when under pressure, rather than cooperate to solve problems together. Furthermore, interviewees described that the many changes resulting from overscoping (E1) were badly communicated to the product unit and resulted in false error reports being filed on changed, but not updated requirements. This in turn, caused irritation among the development teams and further increased the communication gaps. Similarly, Karlsson et al. reported that constant inflow of requirements (cause C1) caused decision conflicts between marketing and development roles [38].
- *challenge to keep SRS updated (E6)*. The majority of the respondents confirmed that overscoping increases the challenge to

Please cite this article in press as: E. Bjarnason et al., Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering, Inform. Softw. Technol. (2012), http://dx.doi.org/10.1016/j.infsof.2012.04.006

keep the SRS updated. When the SRS is detailed upfront (C5), the combination of the two (overscoping) effects E1 (many scope changes) and E1b (decreased motivation) lead to an increased need, but a lower motivation to update the SRS. This complements previous work, which reports requirements volatility as a common challenge for software projects [30,32,34,70] and that the view of RE as concerning a static set of requirements is inappropriate [29,30]. In addition, Berry et al. report that time and resources are never sufficient to keep the documentation updated and that scope creep occurs when programmers code while the documentation keeps changing [9]. Furthermore, our study highlights that the challenge of keeping the SRS updated is increased as an effect of overscoping. Harker and Eason proposed to address this challenge by defining a minimum critical specification combined with incremental deliveries (i.e. +P6) and thereby gradually providing more value [30]. Further research is needed to investigate if the methods proposed to address the challenge of updating the requirements documentation could also minimize this effect for overscoping.

- *overtime (+E7), changed/cancelled product plans (+E8), low priority for administrative tasks (+E9).* These effects were mentioned in the validation questionnaires and each got one count of critical impact. Further investigations are needed to validate their relationship to overscoping.

7.3. How agile RE practices may impact overscoping (RQ3)

Our study identifies that three of the agile RE practices being introduced at the case company may impact several of the causes and root causes of overscoping. In addition, three more practices were suggested by questionnaire respondents as addressing overscoping. The details of how the identified agile RE practices may impact overscoping (mentioned root causes can be seen in Fig. 2) are discussed below. We interpret the results as an indication that overscoping is still a challenge for the case company, though more manageable with the (partly implemented) agile RE practices. Further investigations are needed to fully understand the situation in the agile context.

- *one continuous scope and release planning flow (P1)* is experienced by the responders to directly impact cause C2 (no overview of software resource availability) by enabling transparency and insight into the full project scope and into the current workload of the software unit. The increased visibility of the load and available resource capacity to both business and software unit may bridge several communication gaps identified as root cause of overscoping, i.e. RC1c, RC3d and RC4b. This practice covers the agile RE practices of requirements prioritization and constant re-planning for the high-level requirements [57]. Our results confirm the findings of Dybå and Dingsøyr that managers of agile companies are more satisfied with the way they plan their projects than are plan-based companies [21]. Furthermore, our study also corroborates the findings that agile prioritization of the scope in combination with a stage-gate model at the feature level can avoid delaying critical features and also provides early feedback on features [36]. However, achieving correct high-level cost and schedule estimation has been identified as a challenge also for agile project [57], which may be one reason why overscoping remains an issue for the case company.
- *Cross-functional development teams (P2)* are indicated by our results as improving several of the communication gaps identified by our study as important root causes to overscoping (i.e. RC1c, RC2a, RC3d, RC4b, RC6d). This case company practice is equivalent to the agile RE practice of preferring face-to-face requirements communication over written documentation [8] in combination with agile prioritization and constant re-planning at the detailed requirements level [57]. At this detailed requirements level, cost and schedule estimations in an agile fashion (by only allowing additions when simultaneously removing something less prioritized) have been found to be efficient [36,57] and eliminate the 'requirements cramming' problem [36], which is equivalent to overscoping. Other studies have found that communication within development teams is improved by agile practices, but that communication towards other (dependent) teams remains a challenge [36,53]. This challenge is addressed with P2 by including competence covering all the involved functional areas within the same team (thus, impacting root causes RCicii, RC2a, RC4b and RC6dii). Furthermore, the agile RE practice of including a customer representative in the development teams is summarized by Dybå et al. [21] as improving the communication been customer and engineers, while filling this role can be stressful and challenging [36,57].
- *Gradual and iterative requirements detailing (P3)* is seen (by our interviewees) to decrease the total lead time for development of a feature (root cause RC1b) by delaying the detailing of requirements until they are actually needed for design and development. This in turn reduces the amount of requirement changes within the (shorter) time frame for the feature development, which in a market with high requirements volatility is a significant improvement. It may also reduce the communication gaps that occur due to the timing aspect of detailing requirements before design and implementation starts, i.e. root causes RC3d, RC4a, RC4b. The case company practice P3 is equivalent to the agile practice of iterative RE [57].

7.4. Threats to validity and limitations

As for every study there are limitations that should be discussed and addressed. These threats to validity and steps taken to mitigate them are reported here based on guidelines provided by Robson for flexible design studies [60]. Another important aspect for the quality of a flexible design research is the investigator [60], and for this study all researchers involved have previous experience in conducting empirical research, both interview studies and surveys.

7.4.1. Description validity

Misinterpretation [60] of the interviewees poses the main threat to *description validity*. This threat was addressed in several ways. The interviews were recorded and transcribed. To enhance *reliability* of the transcriptions, the person taking notes during the interviews also transcribed them. In addition, this person has worked for the case company for several years and is well versed in company culture and language. Also, data triangulation was applied to the transcriptions by another researcher performing an independent transcription and coding of two randomly selected interviews. Furthermore, the interviewees checked both the transcriptions and the results of the study for errors and misinterpretations. Finally, data triangulation was applied to the interview results by collecting additional viewpoints from six (other) practitioners through a questionnaire [60].

7.4.2. Interpretation validity

For this study, the main threat to valid interpretation has been the risk of imposing the hypothesis (formulated in phase one) onto the interviewees. To address this threat, open interview questions were always posed before asking specific questions based on the hypothesis. Furthermore, spontaneous descriptions of causes (without prompting) have been reported (as *Experienced*) separately from responses to follow-up questions on specific causes (as *Agreed*), see Section 5.1 and Table 3.

Please cite this article in press as: E. Bjarnason et al., Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering, Inform. Softw. Technol. (2012), http://dx.doi.org/10.1016/j.infsof.2012.04.006

For phase three, the threat to valid description was addressed by the researchers jointly designing the questionnaire and the session held in connection to it. To ensure that all questionnaire responders correctly and uniformly understood the interview results, the results were presented to the participants. They could then ask for clarifications before filling out the questionnaire. The fact that questionnaire responders were confronted with a framework of results remains an open threat to interpretation validity. On the other hand, both interviewees and questionnaire respondents were explicitly encouraged to disagree and mention additional causes, effects and practices, which they also did. One of the main limitations of the study is the limited number of respondents. Although representatives from each of the covered units of the case company were involved in both interviews and validation questionnaire, the number of persons is relatively small and more factors may be identified by including additional viewpoints.

7.4.3. Theory validity

The main threat to theory validity for this study is the risk of missing additional or alternative factors. One source of this threat is the limited set of practitioners from which data has been gathered. Another potential source is the risk of *observer biases* limiting the study to the researcher's pre-knowledge of the company. This was a risk mainly in phase one and was addressed by involving the other researchers in discussing and reviewing the study design and the hypothesis which shaped the interview instrument. The fact that an additional main cause (i.e. C6) was identified as a result of the interviews shows that this bias was successfully addressed. However, identification of additional results in phase 3 may indicate that saturation and the full exploration of the problem under investigation is not yet reached. As the goal of this work is exploratory our aim is not to present or achieve a complete coverage of the problem under investigation.

The involvement of the researcher with work experience from the case company has played a vital role in the study. This has ensured that the investigated problem is authentic and that the results are derived though an interpretation of the data based on a deep understanding of the case and its context. However, the results are limited to the case company and there is a risk that other possible causes of overscoping experienced at other companies were not identified. This also applies to the set of agile RE practices, which are limited to the ones that were currently known and partly implemented at the case company at the time of the study.

Internal generalisability was addressed by sampling interviewees and questionnaire respondents from different parts of the company thereby selecting roles and responsibilities involved throughout the development life cycle. Even so, it was not possible to include representatives from sales and marketing (they were unavailable at the time of the study). However, the requirements team leaders provided some insight into these aspects based on their experience from contacts with customers and with sales and marketing roles.

Considering *external generalisability*, the results should be interpreted with the case company context in mind. External validity is addressed by using *analytical generalization* which enables drawing conclusions without statistical analysis and, under certain conditions, relating them also to other cases [60,62]. Within the scope of this paper, analytical generalization is argued by applying the *making a case* strategy ([60], p. 107) by analysing related work and reporting similarities, differences and disagreements to our results (see Section 7). This analysis builds a supporting argument towards external validity of our study by seeking data which is not confirming a pre-assumed theory. In addition, follow-up studies in other domains can be conducted to utilize the direct demonstration strategy ([60]) to further address the threat to external validity.

8. Conclusions and further work

Decision making is at the heart of requirements engineering (RE) [5] and within market-driven requirements engineering (MDRE) release planning is one of the most important and challenging tasks [38,39,59,65]. Decisions concerning what to develop, and when, are inherently related to achieving customer satisfaction. Even though release planning [37,39,59] is well researched, RE decision making is acknowledged as challenging [2,5,50] and scope creep is ranked as a serious project risk [16,17,31], other aspects of scope management have been less explored [72]. Furthermore, techniques for prioritizing requirements [37,39] often focus on planning the scope of a project as a discrete activity, or one in a series of releases [50]. Our previous work reported that scoping in an MDRE context is a continuous activity that may last throughout the entire project lifecycle [71]. If not successfully managed, and more requirements are included into the project scope than can be handled with available resources the result is *overscoping*, i.e. the project 'bites off more than it can chew'.

Our study provides a detailed picture of factors involved in overscoping and confirms that scoping is a challenging part of requirements engineering and one of the risks in project management [13,20,43]. Our results indicate that overscoping is mainly caused by the fast-moving market-driven domain in which the case company operates, and how this inflow of requirements is managed. In the early project phases, low involvement from the development-near roles in combination with weak awareness of overall goals may result in an unrealistically large project scope. Our study indicates that overscoping can lead to a number of negative effects, including quality issues, delays and failure to meet customer expectations. Delays and quality problems are expensive, not just considering the cost of fixing the quality issues, but also in loss of market shares and brand value [59]. Furthermore, we found indications that a situation of overscoping may cause even more overscoping, i.e. an organization may end up in a vicious cycle when overscoping ties up development resources which are then not available for participating in early project phases. Furthermore, overscoping leads to increased communication gaps, which in turn are root causes of overscoping.

Companies, such as our case company, that develop embedded software for a business domain with a high market pressure need an organizational set-up and process suited to efficiently managing frequent changes in a cost effective way. Development projects need to respond quickly to changes, while at the same time handling the complexity of developing software in a large-scale setting. Agile processes are claimed to be better adapted to managing change than phase-based ones. 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.' Our study indicates, that despite introducing agile RE practices, overscoping is still an issue for the case company, although more manageable. We conclude that the improvements may be explained by the agile RE practices of continuous prioritization of the project scope, in combination with performing cost and schedule estimation, and gradual requirements detailing, in close collaboration within cross-functional teams, thereby closing a number of communication gaps. However, agile RE practices also pose challenges [57], e.g. communication between teams [36,53], difficulty in cost estimation [57]. This, in combination with a fast-moving, market-driven domain may explain why overscoping remains a challenge also with the agile development process.

The causes and effects unveiled through this study (summarized in Fig. 2) can be used as a basis for identifying potential issues to address in order to avoid or alleviate an overscoping situation. For example, the root cause of *low competence in cost estimations*

Please cite this article in press as: E. Bjarnason et al., Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering, Inform. Softw. Technol. (2012), http://dx.doi.org/10.1016/j.infsof.2012.04.006

may be addressed by introducing techniques for improving cost estimation, which should lead to more realistic plans. Finally, supported by our findings of potentially serious effects of overscoping, we conclude that this phenomenon can be a major risk of requirements engineering and project management, complementary to the risk of scope creep mentioned by De Marco and Lister [20].

Future work includes evaluating the agile RE practices when they are fully implemented; how do they affect overscoping and what additional challenges do they pose over time? Furthermore, it would be interesting to investigate how aspects such as organizational set-up, software development model (agile or waterfall) and application of different software engineering methods affect decision making. In addition, extending the results from this study to include other companies and also other perspectives, such as marketing and sales, may strengthen the generalisability of our findings.

Acknowledgements

We would like to thank all anonymous interviewees and questionnaire respondents for their invaluable contribution to this project. We would also like to thank Dr. Dietmar Pfahl for reviewing an early version of this paper. The project is partly funded by the Swedish Foundation for Strategic Research and VINNOVA (The Swedish Governmental Agency for Innovation Systems) within the EASE and UPITER Projects.

References

[1] M. Abramovici, O.C. Sieg, Status and Development Trends of Product Lifecycle Management Systems, Ruhr-University Bochum, Germany, 2002.
[2] B. Alenljung, A. Persson, Portraying the practice of decision-making in requirements engineering: a case of large scale bespoke development, Req. Eng. 13 (2008) 257–279.
[3] A. Aurum, E. Martin, Managing both individual and collective participation in software requirements elicitation process, in: 14th International Symposium on Computer and Information Sciences (ISCIS'99), Kusadasi, Turkey, 1999, pp. 124–131.
[4] A. Aurum, C. Wohlin, Aligning Requirements with Business Objectives: a Framework for Requirements Engineering Decisions, Workshop on Requirements Engineering Decision Support, REDECS'05, 29 August–September 2, 2005, Paris, France.
[5] A. Aurum, C. Wohlin, The fundamental nature of requirements engineering activities as a decision-making process, Inf. Softw. Technol. 45 (2003) 945–954.
[6] A. Aurum, C. Wohlin, Requirements engineering: setting the context, in: A. Aurum, C. Wohlin (Eds.), Managing and Engineering Software Requirements, Springer-Verlag, Germany, 2005, pp. 1–15.
[7] K. Beck, Extreme Programming Explained, Addison-Wesley, 1999.
[8] K. Beck et al., The Agile Manifesto. <http://agilemanifesto.org/> (accessed June 2011).
[9] D.M. Berry, K. Czarnecki, M. Antkiewicz, M. AbdElRazik, Requirements determination is unstoppable: an experience report, in: Proceedings of the 18th International IEEE Requirements Engineering Conference, IEEE Computer Society, 2010, p. 311.
[10] E. Bjarnason, K. Wnuk, B. Regnell, A case study on benefits and side-effects of agile practices in large-scale requirements engineering, in: Proceedings of the 1st Workshop on Agile Requirements Engineering (AREW '11), ACM, New York, NY, USA, 2011.
[11] E. Bjarnason, K. Wnuk, B. Regnell. Overscoping: reasons and consequences – a case study in decision making in software product management, in: Proceedings of the 4th International Workshop on Software Product Management, IEEE Computer Society, September 2010, pp. 30–39.
[12] E. Bjarnason, K. Wnuk, B. Regnell, Requirements are slipping through the gaps – a case study on causes & effects of communication gaps in large-scale software development, in: 19th IEEE International Requirements Engineering Conference, IEEE Computer Society, 2011, pp. 37–46.
[13] B. Boehm, Tutorial: Software Risk Management, IEEE Computer Society Press, 1989.
[14] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, J. Natt och Dag, An industrial survey of requirements interdependencies in software product release planning, in: Proceedings of the Fifth IEEE International Symposium on Requirements Engineering, IEEE Computer Society, August 2001, Toronto, Canada, pp. 84–91.
[15] P. Carlshamre, A Usability Perspective on Requirements Engineering – From Methodology to Product Development, Ph.D. Thesis, Linköping University, Sweden, 2002.
[16] R.A. Carter, A.I. Anton, A. Dagnino, L. Williams, Evolving Beyond requirements creep: a risk-based evolutionary prototyping model, in: Proceedings of the Fifth IEEE International Symposium on Requirements Engineering, Toronto, Canada, August 2001, IEEE Computer Society Press, pp. 84–101.
[17] N. Crockford, An Introduction to Risk Management, second ed., Woodhead-Faulkner, Cambridge, UK, 1980, p. 18.
[18] M.A. Cusumano, R.W. Selby, Microsoft Secrets, Simon and Schuster, New York, 1995.
[19] J.M. DeBaud, K. Schmid, A systematic approach to derive the scope of software product lines, in: Proceedings of the 21st International Conference on Software Engineering, ACM, Los Angeles, USA, 1999, pp. 34–43.
[20] T. DeMarco, T. Lister, Risk Management during Requirements, IEEE Software 20 (2003) 99–101.
[21] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: a systematic review, Inf. Softw. Technol. 50 (2008) 833–859.
[22] C. Ebert, J. De Man, E-R&D – effectively managing process diversity, Ann. Softw. Eng. 14 (2002) 73–91.
[23] K. El Emam, N.H. Madhavji, A field study of requirements engineering practices in information systems development, in: Proceedings of the Second IEEE International Symposium on Requirements Engineering, IEEE Computer Society, Washington, DC, USA, 1995, pp. 68–80.
[24] S.R. Faulk, Product-line requirements specification (PRS): an approach and case study, in: Proceedings of the Fifth IEEE International Symposium on Requirements Engineering, IEEE Computer Society, Washington, DC, USA, 2001, pp. 48–55.
[25] S. Fricker, T. Gorschek, P. Myllyperkiö, Handshaking between software projects and stakeholders using implementation proposals, in: Proceedings of the International Working Conference on Requirements Engineering: Foundation for Software Quality, Trondheim, Norway, 2007, vol. 4542 of Lecture Notes in Computer Science, Springer, Berlin/Heidelberg, 2007, pp. 144–159.
[26] A. Gemmer, Risk management moving beyond process, Computer 30 (1997) 33–43. doi:http://dx.doi.org/10.1109/2.589908 http://dx.doi.org/10.1109/2.589908.
[27] M. Glinz, S. Berner, S. Joos, Object-oriented modeling with ADORA, Inform. Syst. 27 (2002) 425–444.
[28] T. Gorschek, C. Wohlin, Requirements abstraction model, Req. Eng. 11 (2005) 79–101.
[29] T. Hall, S. Beecham, A. Rainer, Requirements problems in twelve software companies: an empirical analysis, IEEE Software 149 (2002) 153–160.
[30] S.D.P. Harker, K.D. Eason, The change and evolution of requirements as challenge to the practice of software engineering, in: Proceedings of the IEEE International Symposium on Requirements Engineering, San Diego, CA, USA, 1993, pp. 266–292.
[31] C.L. Iacovou, A.S. Dexter, Turning around runaway information technology projects, IEEE Eng. Manage. Rev. 3 (2004) 97–112.
[32] C. Hood, S. Wiedemann, S. Fichtinger, U. Pautz, Change management interface, in: Requirements Management – The Interface between Requirements Development and All Other Systems Engineering Processes, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 175–191.
[33] M. Jorgensen, M. Shepperd, A systematic review of software development cost estimation studies, IEEE Trans. Softw. Eng. 33 (2007) 33–53.
[34] P. Jönsson, M. Lindvall, Impact analysis, in: A. Aurum, C. Wohlin (Eds.), Managing and Engineering Software Requirements, Springer-Verlag, Germany, 2005, pp. 117–142.
[35] J. Kabbedijk, S. Brinkkemper, S. Jansen, S.B. van der Veldt, Customer involvement in requirements management: lessons from mass market software development, in: Proceedings of the 17th IEEE International Requirements Engineering Conference, Atlanta, GA, USA, September 2009, pp. 281–286.
[36] D. Karlström, P. Runeson, Combining agile methods with stage-gate project management, IEEE Soft. 22 (2005) 43–49.
[37] J. Karlsson, K. Ryan, A cost-value approach for prioritizing requirements, IEEE Software 14 (1997) 67–74.
[38] L. Karlsson, Å.G. Dahlstedt, J. Natt Och Dag, B. Regnell, A. Persson, Requirements engineering challenges in market-driven software development – an interview study with practitioners, Inf. Softw. Technol. 49 (2007) 588–604.
[39] L. Karlsson, T. Thelin, B. Regnell, P. Berander, C. Wohlin, Pair-wise comparisons versus planning game partitioning–experiments on requirements prioritisation techniques, Empirical Softw. Eng. 12 (2007) 3–33.
[40] M. Khurum, K. Aslam, T. Gorschek, A method for early requirements triage and selection utilizing product strategies, in: Proceedings of the 14th Asia–Pacific Software Engineering Conference, IEEE Computer Society, Washington, DC, USA, 2007, pp. 97–104.
[41] S. Konrad, M. Gall, Requirements engineering in the development of large-scale systems, in: Proceedings of the 16th IEEE International Requirements Engineering Conference, IEEE Computer Society, Washington, DC, USA, 2008, pp. 217–222.
[42] A. Van Lamsweerde, From system goals to software architecture, in: M. Bernardo, P. Inverardi (Eds.), Formal Methods for Software Architectures, Springer, 2003.
[43] I. Legodi, M.L. Barry, The current challenges and status of risk management in enterprise data warehouse projects in South Africa, in: Proceedings of PICMET '10, 18–22 July 2010, pp. 1–5.
[44] T.C. Lethbridge, J. Singer, A. Forward, How software engineers use documentation: the state of the practice, IEEE Software 20 (2003) 35–39.

Please cite this article in press as: E. Bjarnason et al., Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering, Inform. Softw. Technol. (2012), http://dx.doi.org/10.1016/j.infsof.2012.04.006

[45] Z. Lixin, A Project human resource allocation method based on software architecture and social network, in: Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing, October 2008, pp. 1–6.
[46] L. Macaulay, Requirements capture as a cooperative activity, in: Proceedings of the First IEEE Symposium on Requirements Engineering, USA, 1993, pp. 174–181.
[47] N. Medvidovic, P. Grünbacher, A. Egyed, B. Boehm, Bridging models across the software lifecycle, J. Syst. Softw. 68 (2003) 199–215.
[48] M.D. Myers, D. Avison, Qualitative Research in Information Systems, Sage Publications, USA, 2002.
[49] L. Neumann-Alkier, Think globally, act locally – does it follow the rule in multinational corporations? in: Proceedings of the Fifth European Conference on Information Systems, 1997, pp. 541–552.
[50] A. Ngo-The, G. Ruhe, Decision support in requirements engineering, in: A. Aurum, C. Wohlin (Eds.), Managing and Engineering Software Requirements, Springer-Verlag, Germany, 2005, pp. 267–286.
[51] B. Nuseibeh, Weaving together requirements and architectures, Computer 34 (2001) 115–117.
[52] J.J. Phillips, T.W. Bothell, G.L. Snead, The Project Management Scorecard: Measuring the Success of Project Management Solutions, Elsevier, USA, 2002.
[53] M. Pikkarainen, J. Haikara, O. Salo, P. Abrahamsson, J. Still, The impact of agile practices on communication in software development, Empirical Softw. Eng. 13 (2008) 303–337.
[54] C. Pohl, G. Böckle, F.J. van der Linden, Software Product Line Engineering: Foundations, Principles and Techniques, Springer-Verlag, New York, USA, 2005.
[55] C. Potts, Invented requirements and imagined customers: requirements engineering for off-the-shelf software, in: Proceedings of the Second IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, March 1995, pp. 128–131, doi: http://dx.doi.org/10.1109/ISRE.1995.512553.
[56] Project Management Institute, A Guide to the Project Management Body of Knowledge (PMBOK Guide), 2000 Ed., Project Scope Management. Project Management Institute, Four Campus Boulevard, Newtown Square, PA 19073–3299, USA, 2000 (Chapter 5).
[57] B. Ramesh, L. Cao, R. Baskerville, Agile requirements engineering practices and challenges: an empirical study, Inform. Syst. J. 20 (2007) 449–480.
[58] B. Regnell, R. Berntsson-Svensson, K. Wnuk, Can we beat the complexity of very large-scale requirements engineering?, in: B. Paech, C. Rolland (Eds.), Proceedings of the 14th International conference on Requirements Engineering: Foundation for Software Quality (REFSQ '08), vol. 5025 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 123–128.
[59] B. Regnell, S. Brinkkemper, Market-driven requirements engineering for software products, in: A. Aurum, C. Wohlin (Eds.), Managing and Engineering Software Requirements, Springer-Verlag, Germany, 2005, pp. 287–308.
[60] C. Robson, Real World Research, Blackwell Publishing, 2002.
[61] D. Rosca, S. GreenSpan, M. Feblowitz, C. Wild, A decision making methodology in support of the business rules lifecycle, in: Proceedings of the Third IEEE International Symposium on Requirements Engineering, Annapolis, MD, USA, January 1997, pp. 236–246.
[62] P. Runeson, A. Rainer, M. Höst, B. Regnell, Case Study Research in Software Engineering: Guidelines and Examples, Wiley, 2012.
[63] R. Sangwan, M. Bass, N. Mullick, D.J. Paulish, J. Kazmeier, Global Software Development Handbook, Auerbacj Publications, Boston, MA, USA, 2006.
[64] P. Sawyer, Packaged software: challenges for RE, in: Proceedings of the 6th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'2000), Stockholm, June 2000.
[65] P. Sawyer, I. Sommerville, G. Kotonya, Improving market-driven re processes, in: Proceedings of the International Conference on Product-Focused Software Process Improvement, Oulu, Finland, June 1999, pp. 222–236.
[66] K. Schmid, A comprehensive product line scoping approach and its validation, in: Proceedings of the 24th International Conference on Software Engineering, IEEE Computer Society, Orlando, USA, May 19–25 2002, pp. 593–603.
[67] K. Schwaber, M. Beedle, Agile Software Development with SCRUM, Prentice Hall, 2002.
[68] M. Svahnberg, T. Gorschek, R. Feldt, R. Torkar, S. Bin Saleem, M.U. Shafique, A systematic review on strategic release planning models, Inf. Softw. Technol. 52 (2010) 237–248.
[69] Case Study Material (interview instrument, questionnaire, etc) for the Before aNd After (BNA) study. <http://serg.cs.lth.se/research/experiment_packages/bna/>.
[70] K.E. Wiegers, Software Requirements, second ed., Microsoft Press, Redmond, 2003.
[71] K. Wnuk, B. Regnell, L. Karlsson, What happened to our features? visualization and understanding of scope change dynamics in a large-scale industrial setting, in: Proceedings of the 17th IEEE International Requirements Engineering Conference, IEEE Computer Society Press, September 2009, Atlanta, GA, USA, pp. 89–98, doi: http://dx.doi.org/10.1109/RE.2009.32.
[72] I. van de Weerd, S. Brinkkemper, R. Nieuwenhuis, J. Versendaal, L. Bijlsma, Towards a reference framework for software product management, requirements engineering, in: 14th IEEE Internal Conference, September 2006, pp. 319–322.
[73] C. Wohlin, A. Aurum. What is important when deciding to include a software requirement in a project or release?, in: 4th Symposium on Empirical, Software Engineering, 17–18 November 2005, doi: http://dx.doi.org/10.1109/ISESE.2005.1541833.

Please cite this article in press as: E. Bjarnason et al., Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering, Inform. Softw. Technol. (2012), http://dx.doi.org/10.1016/j.infsof.2012.04.006