What Happened to Our Features? Visualization and Understanding of Scope Change Dynamics in a Large-Scale Industrial Setting

Krzysztof Wnuk[1], Björn Regnell[1], Lena Karlsson[2]
[1]{krzysztof.wnuk,bjorn.regnell}@cs.lth.se, Lund University, Sweden
[2] lena.karlsson@dnv.com, Det Norske Veritas, Sweden

Abstract

When developing software platforms for product lines, decisions on which features to implement are affected by factors such as changing markets and evolving technologies. Effective scoping thus requires continuous assessment of how changes in the domain impact scoping decisions. Decisions may have to be changed as circumstances change, resulting in a dynamic evolution of the scope of software asset investments. This paper presents an industrial case study in a large-scale setting where a technique called Feature Survival Charts for visualization of scoping change dynamics has been implemented and evaluated in three projects. The evaluation demonstrated that the charts can effectively focus investigations of reasons behind scoping decisions, valuable for future process improvements. A set of scoping measurements is also proposed, analyzed theoretically and evaluated empirically with data from the cases. The conclusions by the case company practitioners are positive, and the solution is integrated with their current requirements engineering measurement process.

1. Introduction

Deciding which requirements to include into the scope of an upcoming project is not a trivial task. Requirements for complex systems may be counted in thousands, and not all may be included in the next development project or next release. This means that it is necessary to select a subset of requirements to implement in the forthcoming project, and hence postpone the implementation of other requirements to a later point in time [1, 12]. This selection process is often called *scoping* and is considered as a key activity for achieving economic benefits in product line development [2]. While its importance has already been reported in several studies, research has not yet put broad attention to the issues of product line scoping. In particular, following Schmid [2], we agree that existing work in domain engineering in software product lines focus mainly on the *identification aspect* of scoping e.g. [7, 13]. On the other hand, some researchers have already addressed the issue of understanding underlying reasons for the inclusion of certain requirements in a specific release [1], while others investigated one of the root causes for changing requirements, namely requirements uncertainty [14].

The problem with many changes in the scoping process for product line projects has recently been identified by one of our industrial partners from the embedded systems domain. This issue has been particularly challenging for the case company, since their current requirements management tool could not provide a sufficient method to visualize and characterize this phenomena. As a remedy, the Feature Survival Chart (FSC) concept was proposed by the authors and acknowledged by the practitioners as a valuable support. This paper extends the contributions of [3] with (1) findings from industrial application in three projects and (2) scope tracking measurements. The proposed visualization shows the decision process of including or excluding features that are candidates for the next release. Our technique can spot the problem of setting too large a scope compared to available resources as well as increase the understanding of the consequences of setting a limited scope early. By using graphs, we can identify which features and which time frames to analyze in order to find scoping issues related to uncertainties in the estimations that decisions rely on. The charts have also shown to be useful in finding instabilities of the scoping process.

The proposed set of scope tracking measurements complements the proposed visualization technique, and they aim at further increasing the understanding of the rationale and dynamics of scope changes. The measurements are analyzed both theoretically and empirically using data from three large industrial projects that contain hundreds of high-level features

1090-705X/09 $25.00 © 2009 IEEE
DOI 10.1109/RE.2009.32

related to thousands of system requirements. We also present findings from discussions on the results with practitioners that ranked the usefulness of the proposed measurements and expressed their opinions about their value in scope management.

The paper is structured as follows: Section 2 provides background information about the context of our industrial case study. Section 3 describes the methodology used in this study. Section 4 explains our visualization technique. Section 5 describes the results from applying our technique to three industrial projects. Section 6 defines and evaluates the proposed measurements. Section 7 provide conclusions and discusses their limitations.

2. The case company

Our results are based on empirical data from industrial projects at a large company that is using a product line approach [4]. The company has more than 5000 employees and develops embedded systems for a global market. There are several consecutive releases of the platform, a common code base of the product line, where each of them is the basis for one or more products that reuse the platform's functionality and qualities. A major platform release has approximately a two year lead time from start to launch, and is focused on functionality growth and quality enhancements for a product portfolio. Minor platform releases are usually focused on the platform's adaptations to the different products that will be launched with different platform releases. This approach creates an additional requirements flow, which in our case company is handled as a *secondary flow,* and arrives to the platform project usually in the middle of its life cycle. This flow enables flexibility and adaptation possibilities of the platform project, while the *primary flow* is dedicated to address functionality of the highest importance.

There are several groups of specialists associated with various stages of the requirements management process in the case company. For this case, the most essential groups are called *Requirements Teams (RTs)* that elicit and specify high-level requirements for a special technical area, and *Design Teams (DTs)* that design and develop previously defined functionality.

The company uses a stage-gate model with several increments [5]. There are *Milestones (MSs)* and *Tollgates (TGs)* to control the project progress. In particular, there are four milestones for the requirements management and design before the implementation starts: MS1, MS2, MS3, and MS4. For each of these milestones, the project scope is updated and baselined. The milestone criteria are as follows:

MS1: At the beginning of each project, long-term RT's roadmap documents are extracted to formulate a set of features for an upcoming platform project. A *feature* in this case is a concept of grouping requirements that constitute a new functional enhancement to the platform. At this stage the features usually contain a description, its market value and effort estimates. The level of details for the features should be set up in a way that enables judgment of its market value and effort of implementation. Both values are obtained using a cost-value approach [6]. The cost for implementation and the market value of features are the basis for initial scoping inclusion for each technical area. The features are reviewed, prioritized and approved. The initial scope is decided and baselined per RT, guided by a project directive and based on initial resource estimates in the primary receiving DT. The scope is then maintained in a document called *Feature List,* that is regularly updated each week after a meeting of the *Change Control Board (CCB).* The role of the CCB is to decide upon adding or removing features according to changes that happen. The history of scope changes is the input data for the visualization technique described in this paper.

MS2: Features are refined to requirements which are specified, reviewed and approved. One feature usually contains ten or more requirements from various areas in the products. The features are assigned to DTs that will take responsibility for designing and implementing the assigned features after MS2. The DTs also allocate an effort estimate per feature.

MS3: The effort estimates are refined and the scope is updated and baselined. DTs refine system requirements and start designing.

MS4: The requirements work and design are finished, and ready to start implementation. The final scope is decided and agreed with the development resources.

According to the company guidelines, most of the scoping work should be done before reaching the second milestone of the process. The secondary flow starts approximately at MS2 and is connected to the start of product projects. Both primary and secondary flows run in parallel under the same MS criteria until they are merged together when the secondary flow reaches its MS4. The requirements are written in domain-specific natural language, and contain many special terms that require contextual knowledge to be understood. In the early phases, requirements contain a high-level customer-oriented description while being refined to detailed implementation requirements at a late stage.

3. Research methodology

The development of the FSC chart and corresponding scope tracking measures was performed in an interactive manner that involved practitioners from the case company. The persons that participated in the constant evolution and evaluation include one process manager, two requirements managers and one KPI (Key Performance Indicators) manager. This approach involves a set of meetings and discussion points between the researchers and the practitioners that helped to guide the research. As a part of the discussion, the important need to measure the dynamics and the nature of the scope changes emerged. After proposing and theoretically validating the measurements, it was decided to apply them to the real scoping data to empirically confirm the perceived usefulness of the metrics. All ongoing projects in the case company were investigated for possible usage of our technique. Our criteria of interest in analyzing a particular project include (1) the length of analyzed project, (2) the number of features considered in the scope of the project and (3) the possibility to visualize and analyze significant scope changes in the analyzed project. As a result, the three most interesting ones were selected. Furthermore, we have used our technique to define a set of scoping quality measurements that we evaluated by practitioners and validated using empirical data. Finally, we have performed an interview study with platform project requirements managers in order to understand the rationale and implications for scoping decisions.

To gather data for this study, we have implemented an exporter to retrieve the data from the scope parameter of each feature in the *Feature List* document. This information was later sorted so that each feature is mapped into one row and each value of the scope attribute is mapped to an integer value. After creating graphs, a meeting with practitioners was held in order to present and discuss results as well as address issues for future work. As a result of this meeting, it was decided to introduce and evaluate a set of scope tracking measurements that may give a better insight into the scoping process practices and may help to assess their quality. As one of the measurements, it was decided to include a non-numerical reason for scope exclusion to understand their nature and implications on the stability of the requirements

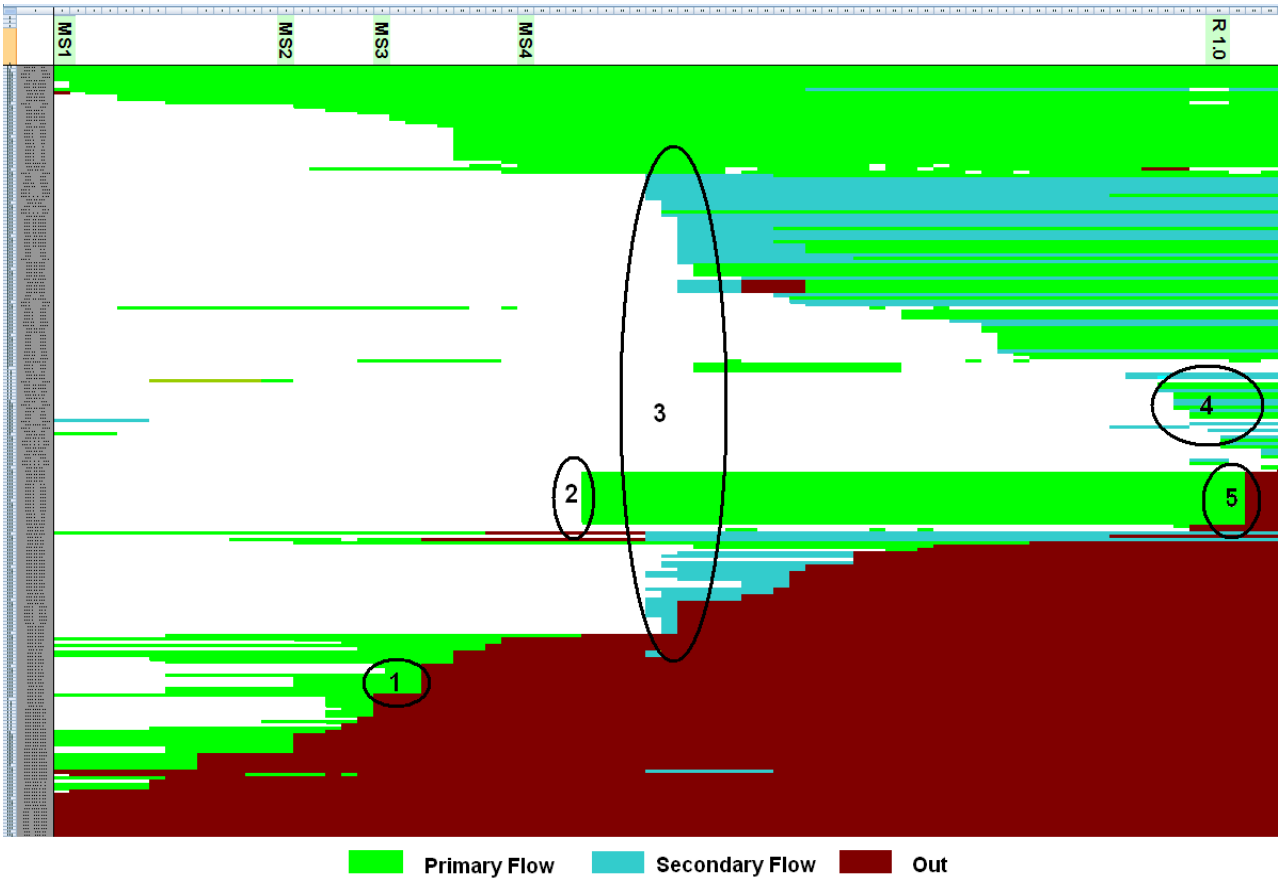

Figure 1 - Feature Survival Chart for project A.

management process. All measurements were calculated on an industrial set of three large platform projects.

4. Feature Survival Charts

In this section, we briefly describe our visualization technique. The *Feature Survival Chart* (FSC), exampled in Figure 1, shows scope changes over time which is illustrated on the X-axis. Each feature is positioned on a specific place on the Y-axis so that the complete lifecycle of a single feature can be followed by looking at the same Y-axis position over time. The various scope changes are visualized using different colors. As a result, each scope change can be viewed as a change of the color. Based on discussions with practitioners we decided to use this coloring scheme: green for features considered as a part of the scope, red for features considered as de-scoped and, if applicable, different shades of green for primary and secondary flows. After sorting the features according to how long they were present in the scope, we get a graph where several simultaneous scope changes can be seen as 'steps' with areas of different colors. The larger the red areas are, the more features are de-scoped in the particular time of the project. At the top of the graph we can see features that we called 'survivors'. These features represent functionality that was included early while lasting until the end of the scoping process. An FSC is also visualizing overall trends in scoping. In Figure 1 we can see that most of scoping activity happened after MS2 in the project. (Rn.m denotes formal releases.) Since most of the de-scoping was done rather late in the project, we can assume that a significant amount of effort might have been spent on features that did not survive. Thanks to the graphs, we can see which decisions have been made when and how large impact on the scope they had. The five areas marked in Figure 1 are further discussed in section 6.3.5. The FSC gives a starting point for investigating why the decisions were made, and enables definition of measurements that indicate quality aspects of the scoping process.

5. Evaluation results

In this section, we present results from evaluating our visualization technique. We present FSCs for three large platform projects in the case company. The data was gathered during autumn 2008 when all three projects were running in parallel and were targeted for product releases in 2008 or 2009. Each project was started at different points in time. At the time when this study was performed one of them had already passed MS4, one had launched the first platform release and the third had passed MS3. In Figure 1, 2 and 3 we present one FSC respectively for three projects denoted A, B and C. Additional information about the projects is presented in Table 1.

All analyzed projects have more than 100 features ever considered in the scope. For projects B and C, the significant feature number difference is a result of running these two projects in parallel targeted to be released the same year. The technical areas are similar for all projects. We can assume that the projects affect similar groups of requirements analysts, but differ in size, time of analysis and complexity. Project A was analyzed during a time period of 77 weeks, during which period two releases of the platform were launched. The total number of scope changes in the projects is calculated from MSA and onwards.

Project	Nbr. of features	Nbr. Of Technical areas	Time Length (weeks)	Total number of scope changes
A	223	22	77	237
B	531	23	39	807
C	174	20	20	43

Table 1 - Characteristics of analyzed projects

Results indicate that we in average experience almost one scope decision per feature for each project. This fact indicates the need for a better understanding of the scoping process, e.g. by visualizing scope changes. A qualitative analysis of the graphs indicates that for all analyzed projects the dominant trend is de-scoping rather than scope increases. We name this phenomena *negative scoping* . For all analyzed projects we can observe negative scoping all through the analyzed period.

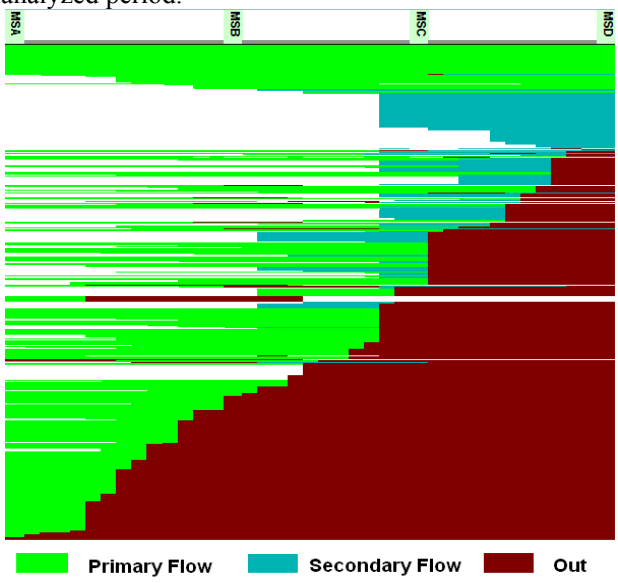

Figure 2 - FSC for project B.

6. Scope tracking measurements

According to Basili et al. [8], measurement is an effective mechanism for characterizing, evaluating, predicting and providing motivation for the various aspects of software construction processes. The same author states that most aspects of software processes and products are too complicated to be captured by a single metric. Following this thread, we have formulated questions related to external attributes of the scoping process, which in turn is related to internal

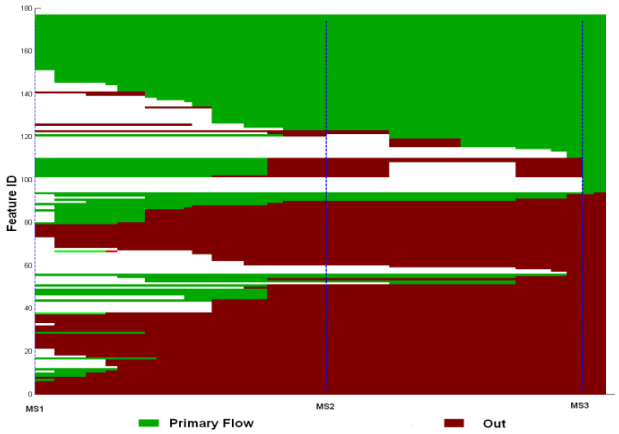

Figure 3 - FSC for project C.

attributes and a set of five measurements divided into time related measurements and feature related measurements, as described subsequently.

6.1. Definition of measurements

The goal with the measurements is to characterize volatility and velocity of the scoping process, as well as clarity of the reasons behind them. To address this goal, we have defined a set of five scope tracking measurements, which are presented subsequently. Four out of five measurements can be calculated based on the scope attribute value in the feature list document and time stamps for this document, while the last measurement needs a more qualitative approach that requires ing additional information that complements the graphs.

6.1.1 Time-related scope tracking measurements: In this category we have defined one measurement:
Number of positive and negative scope changes per time stamp/baseline (M1). We define a positive scope change as an inclusion of a feature into the scope of the project, while a negative change indicates exclusion from the project. We assume that the scope ideally would stabilize in the late phase of the project in order to avoid expensive late changes.

6.1.2 Feature related measurements: In this category we have defined the following measurements that also can be averaged for the whole platform project:
Time to feature removal (M2) – the time from the feature was introduced until it was permanently removed. The measurement can of course only be calculated for the features that have not survived until the end of the requirements management process. The interpretation of this measurement can be as follows: it is a matter of quality of the requirements management process to remove features that will not be included into the projects due to various reasons as early as possible. This approach saves more resources for the features that will be included into the scope, and increases the efficiency of the scoping process. The pitfall related to this measurement is the uncertainty whether features included into the scope at the end of the requirements management process will not be excluded later due to various reasons. On the other hand, even taking this fact into consideration, we still believe that we successfully can measure M2 and get valuable indications of the final scope crystallization abilities.
Number of state changes per feature (M3)– this measurement is a reflection of the measurement M1. By calculating this measurement for all features and visualizing results in the form of a distribution, we can see the fraction of complex decisions among all decisions. The interpretation of this measurement is that the fewer changes per feature in a project, the more 'stable' the decision process is and less extra effort has to be spent on complex decisions making the project less expensive to manage. As already mentioned, high values for this measurement indicate complex and frequently altered decisions.
Time to birth (M4) – for each feature that has not yet appeared in the scope, we calculate the delay time which is proportional to the number of baselines of the scope document. In our calculations, we took into consideration the fact the feature list document was baselined irregularly, and we based our calculations on the number of days between the baselines. This measurement describes the activity of the flow of new features in time. Here, similarly to M1, we have to decide what is our starting point in the project. Our interpretation assumes that we take MS1 as a starting point. In an ideal situation we expect few features with a long time to birth, since late additions to the scope create turbulence in the project.
Reason for scoping decision (M5). As the last measurement described in this study, we define reasons for scoping decisions. This measurement will be calculated as a non-numerical value and it can not be automatically derived from our graphs. As already mentioned in M1, inclusion of a new functionality is a

different change compared with an exclusion of a functionality. Due to limited access to practitioners, we focused on analyzing removal of functionality. To calculate M5, we mapped each feature to its reason for inclusion, reason for exclusion and existing CCB records.

6.2. Theoretical analysis of measurements

In this section, we present results from a theoretical analysis of the proposed measurements. We have used two approaches: "key stage of formal measurement" [9] and the theoretical validation [10]. By following the key stages of formal measurement, we constructed empirical and mathematical systems and defined a mapping between these two systems. The attributes of an entity can have different measurements associated to them, and each measurement can be connected to different units. Some properties, for example mapping between real world entities to numbers and the fact that different entities can have the same attribute value, are by intuition satisfied for all defined measurements. In Table 2 we present defined attributes and relations. We also relate defined measurements with internal and external attributes of the requirements decision process. As we can see in Table 2, the defined set of measurements is addressing stability, velocity, volatility and understandability of the scoping process for platform projects. Although four out of five defined measurements are realized as objective numbers, conclusions drawn from them about subjective attributes of requirements management decision process are a matter of interpretation. The subjective interpretation of the results derived by our measurements is a complex task which requires a deep domain knowledge and additional information about the history of the project. We have extended our knowledge by interacting with requirements managers working with platform projects in order to derive values for M5.

Measurement	Entity	Internal attribute	External attribute	Measure	Domain	Scale	Empirical relation	Mathematical relation
M1	Feature List	Size and direction of scope changes over time.	Stability of the scoping process	# of scope inclusions at the timestamp # of scope exclusions at the timestamp	Feature List	Ratio	negative, positive, bigger smaller, equal to, addition, subtraction, division	<,>,=,+,-, etc.
M2	Feature	The time that was needed to remove the feature from the scope	Velocity of the final scope crystallization process	# days needed to make a final decision about feature exclusion	Feature	Ratio	bigger, smaller, equal to, addition, subtraction, division	<,>,=,+,-, etc.
M3	Feature	Number of scope decisions per feature	Volatility and dynamics of the scope decisions.	#scope changes for non-survivors needed to remove them from the scope.	Feature	Ratio	bigger, smaller, equal to, addition, subtraction, division	<,>,=,+,-, etc.
M4	Feature	Time when a feature was included into the scope of the project	Volatility of the scope decisions.	# of days from the beginning of the project until a feature was included	Feature	Ratio	bigger, smaller, equal to, addition, subtraction, division	<,>,=,+,-, etc.
M5	Changes to feature	Rationale for removing features from the scope	Clarity of the reasons for scope decisions	Reasons for scope exclusions	Scope changes	Nominal	equal and different	<>,=

Table 2 - Results from a theoretical analysis of proposed measurements, by # we mean 'number of'

6.3. Empirical application of measurements

In this section, we present results from an empirical evaluation of measurements defined in section 6.1. We have evaluated M1-M4 in three large platform projects described in section 5, and M5 in one large project. To increase the possibilities of drawing conclusions, we have decided to present time-related measurements as a function of time, while feature-related measurements are presented in the form of distributions for each evaluated project.

6.3.1. Number of positive and negative scope changes per time stamp/baseline (M1). All three projects turned out to have many scope changes over time. In Figure 4 we can see many fluctuations of M1 values both on the positive and negative side rather late in analyzed projects. This result can be explained by a

stage-gate model for requirements management projects resulting in high peaks of changes around project milestones. On the other hand, we experience more than four peaks for each project, which is more than the number of milestones in the requirements management process. The distinction of positive and negative changes makes it possible to see in Figure 4 that inclusions of new functionality into the project may be correlated with exclusions of some other functionality. The baseline number represents the version of the scope document. The best example is the peak of inclusions for Project A around baseline 38, which immediately resulted in a peak of exclusions. In this example we can also see that the magnitude of the change in both directions is similar.

6.3.2. Time to remove a feature (M2). For this measurement, we present results in the form of distributions. The distribution presented in Figure 5 is showing that many features were removed after a certain number of days in the scope. Our results reveal three different approaches for removing the features from the scope. For project A we can see an initial scope reduction rather early, then a quite constant number of

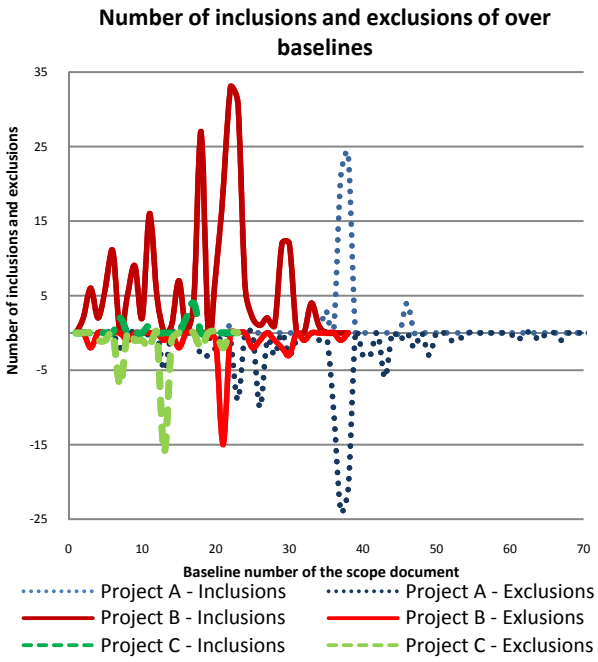

Figure 4 - Number of positive and negative changes as a function of a baseline number (M1).

removed features, and suddenly, after about 300 days from the project start, large scope reductions. For project B we can see that many features were removed in rather short intervals in time, and also that some significant scope reductions that occurred after 150 days in the project. On the other hand, project C is behaving more stable in this matter, having only one large peak of removed features around 60 days from the project launch. This type of graph can be useful in assessing how good the process is in crystallizing the final scope of the platform project.

6.3.3 Number of state changes per feature (M3). For this measurement, we present the results in the form of distributions. As we can see in Figure 6, most features required only one decision in the project. This decision usually was an exclusion from the project scope, but in some cases more than one decision per project was needed. This fact indicates that features were shifted between the primary and the secondary flow of requirements, or that the management had to reconsider previously made commitments. For a better understanding of more complex decisions, this measurement can be limited to the number of scope changes needed to remove the feature from the scope of the project. This measurement may give valuable insights about the complexity of decision-making. We have calculated a derived measurement, and the results are available online [11].

6.3.4 Time to birth (M4). Empirical application of M4 presented as a distribution over time revealed that some projects have a large peak of new functionality coming into the scope of the project after 100 days from the beginning. In two out of three analyzed cases we experienced large scope extensions at various points in the project timeline. The biggest limitation of this measurement is the fact that the used process allows for a secondary flow of requirements which automatically can create large peaks of births at a certain time. We can notice this fact in Figure 7 as a peak of births around day 150 day for project B, and around day 220 for project A. Although the mentioned

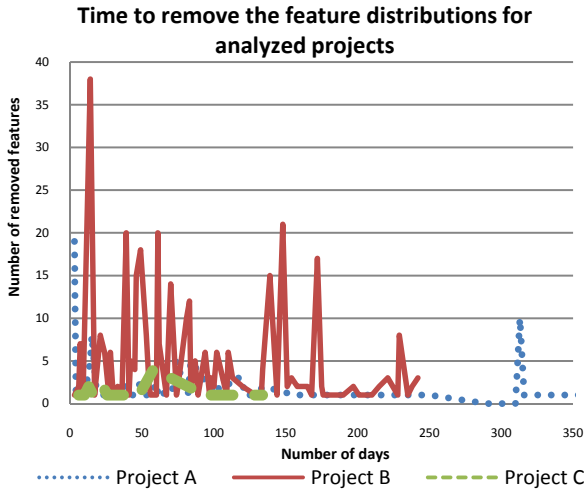

Figure 5 – Distributions of time to remove the feature (M2).

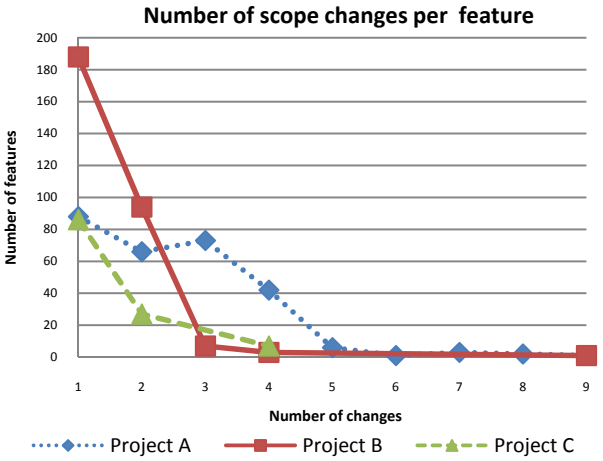

Figure 6 - Distribution of number of changes per feature (M3).

peaks are not necessarily revealing any unplanned behavior, Figure 7 reveals that smaller but still significant scope inclusions appeared for project B both before and after the biggest peak of incoming features.

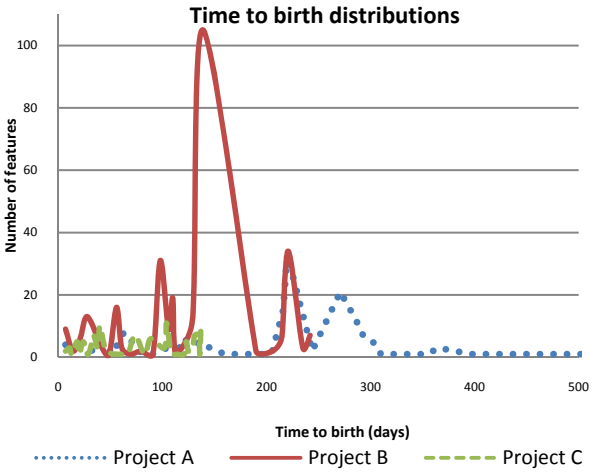

Figure 7 - Time to birth distributions (M4).

6.3.5 Reason for scoping decision (M5). Since M5 is defined as a non-numerical measurement in order to apply it to our industrial data set and gather results, we held a meeting with two requirements managers responsible for managing project scoping information. Each of the requirements managers was responsible for one scoping project. In this paper, we focus on project A since it was the most interesting in terms of late scope changes. Before the meeting, we prepared five scope-zones which we assumed to be the most interesting to analyze, see Figure 1. During the interview, a responsible requirements manager checked the reasons for a particular scope change. The reasons were analyzed both per individual feature, as well as per set of changes in order to identify possible dependencies between various decisions.

Results for scope changes for project A. As we can see in Figure 1, we decided to include changes from both before and after MS4. The results are presented below:

Zone 1 – A significant scope reduction after MS3: This zone shows a large scope reduction that happened between MS3 and MS4 in the platform project. The analysis revealed that this zone includes two reasons for de-scoping. The first one is the strategic reason and the other one is the cancellation of one of the products from the product line project.

Zone 2 – A large scope inclusion after MS4: This zone shows a large set of features introduced to the scope of the project after MS4. The reasons for this change turned out to be an ongoing work to improve performance requirements. Because of this reason, it was decided shortly after MS4 to include these features into the scope.

Zone 3 – A large scope inclusion together with a parallel scope exclusion: This zone represents a desired behavior of the process used in the company. The large scope inclusions show a new flow of requirements related to one of the platform releases. Our responders confirmed that all three sets of features, separated from each other on the graph, represent an introduction of a new requirements flow. The focus for the analysis in this case was to examine if there was any relation between inclusion of new requirements and exclusion of other requirements at the same time. The set of de-scoped features turned out not to be related to the big scope inclusion. As described by the interviewed requirements manager, the main reasons for these scope changes were defined as "*stakeholder business decision*", which means that the previously defined plan was changed to accommodate other aspects of the product portfolio.

Zone 4 – Some incremental scope inclusions introduced very late in the project: As we can see in Figure 1, this zone covers many of the incremental scope inclusions by the end of the analyzed time. Since late scope extensions may put reliability at risk, we investigated why they occurred and found out that there are many reasons behind this phenomena. One of the large changes, that involved four features, was caused by administrative changes in the requirements database. Some additional five features were included into the scope as a result of a late product gap analysis. A gap analysis is a task that requirements managers perform in order to ensure that the scheduled product features are covered by the corresponding platform project. Finally, seven features introduced into the

scope turned out to be a result of late negotiations with one of the customers.

Zone 5 – Late removal of previously accepted features: In this zone, we analyze removal of the features that were analyzed in zone 2. We have asked our responders why initially accepted features later were de-scoped. They replied that despite these features were initially approved, a new decision had to be made mainly due to a lack of available development resources. We also performed a quantitative analysis of reasons for de-scoping in project A. The results are presented in Figure 8. We have analyzed 120 de-scoping decisions that belong to project A. The result is shown in percentages in Figure 8, summing up to 100%. As we can see, 33% of the de-scoping decision were caused by a stakeholder's business decision, and 29% by a lack of resources, while 9% of the decisions were caused by changes in product portfolios. Our largest category, stakeholder business decision is similar to the category mentioned by Wohlin et al. [1]. called "Stakeholder priority of requirement".

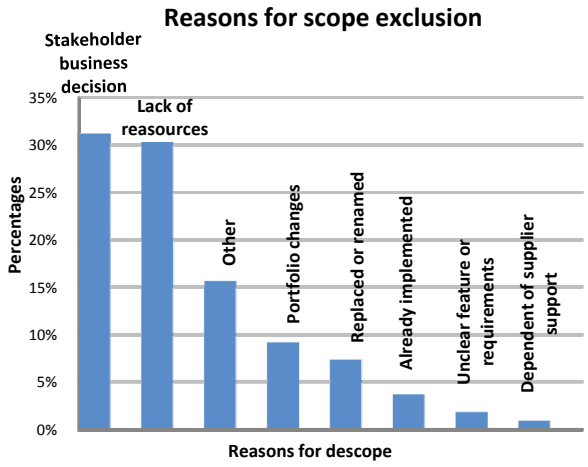

Figure 8 - Quantitative analysis of reasons for removing features from the scope of the Project A.

Therefore we can assume that the dominant reason for both inclusions and exclusion of certain requirements in a specific release does not differ significantly. Furthermore, criteria such as requirements volatility and resource availability seem to appear both in our study and in [1].

Rank	Responder 1	Responder 2	Responder 3
1	M2	M2	M5
2	M5	M5	M1
3	M1	M4	M2
4	M4	M1	M4
5	M3	M3	M3

Table 3 – Results from practitioners' ranking of proposed measurements.

As an additional validation step, we asked three practitioners working with scoping to rank the proposed measurements. As a criterion for ranking, we chose usefulness in understanding the scoping processes and in defining future improvements. The measurement ranked as number one is considered to be the most useful one, while the one ranked in position five is the least useful one. The results are presented in Table 3. As we can see in Table 3, M3 was ranked as the least useful, while M2 and M5 were placed in the top three positions for all responders.

7. Conclusions

According to Basili et al. [8], software engineers and managers need real-time feedback in order to improve construction processes and products in ongoing projects. In the same manner, the organization can use post mortem feedback in order to improve the processes of future projects [17]. Furthermore, visualization techniques used in software engineering have already proven to amplify human cognition in data intensive applications, and support essential work tasks [15]. Our visualization technique provides feedback about ongoing scoping activities as well as a visualization of past project scoping activities. Measurements presented in this paper are complementing our visualization technique by quantitative characterization and qualitative rationale for scoping decisions. The results in terms of usefulness of the proposed visualization technique and scope tracking measurements were acknowledged by practitioners involved in their development as valuable since they confirm the volatility of the scope and provide a tool to analyze the various aspects of this phenomenon. The results were then used by the case company to adjust the process towards more flexibility in scope setting decisions, and a clearer scope responsibility. Our solution has confirmed to outperform the previously used table-based textual method to track the scope changes in the case company. It gives a better overview of the scoping process of the whole project on a single page size graph. The industrial evaluation has indicated that our method can be applied to large scale projects, which demonstrates the scalability of the method. Finally, the managers at the case company decided that our visualization technique should be implemented as a standard practice and is currently in widespread usage at the case company. Even if the characteristics of scope changes found may be particular to this case study, we believe that the manner in which these graphs together with measurements are used to

increase the understanding of the performance of the scoping process is generally applicable.

Limitations. As for any empirical study, there are threats to the validity. One threat is related to the mapping between measurements and external attributes. In software engineering we often want to make a statement of an external attribute of an object. Unfortunately, the external attributes are mostly indirect measurements and they must be derived from internal attributes of the object [16]. We are aware that our mapping can be one of several possible mappings between internal and external attributes. We address its correctness by evaluating external attributes with practitioners in the case company. Another threat is related to the generalization of our results. Although the company is large and develops technically complex products, it cannot be taken as a representative for all types of large companies and hence the results should be interpreted with some caution. Finally, theoretical validation is context dependent and thus needs to be redone in every new context.

Further work. Additional studies of scope dynamics visualization in other cases would further increase our understanding of their usefulness. Enhanced tool support with the possibility of zooming interactively may be useful, as well as depiction of size and complexity of features by visualizing their relation to the underlying system requirements. How to optimize usability of such a tool support, and the search for new possibilities while observing practitioners using the visualization techniques, are also interesting matters of further research.

Acknowledgements. This work is supported by VINNOVA (Swedish Agency for Innovation Systems) within the UPITER project. Special acknowledgements to Thomas Olsson for valuable contributions on scope tracking measurements and to Lars Nilsson for valuable language comments.

8. References

[1] C. Wohlin, A. Aurum, "What is Important when Deciding to Include a Software Requirements in a Project or Release?", *2005 International Symposium on Empirical Software Engineering, ISESE 2005,* Institute of Electrical and Electronics Engineers Computer Society, Piscataway, NJ 08855-1331, United States, 2005, pp. 246-255.

[2] K. Schmid, "A Comprehensive Product Line Scoping Approach and Its Validation", *24th International Conference on Software Engineering (ICSE 2002),* Institute of Electrical and Electronics Engineers Computer Society, Orlando, FL, United States, May 19-25 2002, pp. 593-603.

[3] K. Wnuk, B. Regnell, L. Karlsson, " Visualization of Feature Survival in Platform-Based Embedded Systems Development for Improved Understanding of Scope Dynamics", *Third International Workshop on Requirements Engineering Visualization (REV'08)*, Barcelona, 8th September 2008.

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

[5] R.G. Cooper, "Stage-Gate Systems: A New Tool for Managing New Products", *Business Horizons,* May-June 1990, pp. 44-54.

[6] J. Karlsson, K. Ryan , "Cost-value approach for prioritizing requirements", *IEEE Software,* IEEE, Los Alamos, Sept-Oct 1997, pp. 67-74.

[7] T. Kishi, N. Noda and T. Katayama, "A Method for Product Line Scoping Based on Decision-Making Framework", *Proceeding Second International Conference, SPLC 2002,* Springer Berlin / Heidelberg, San Diego, CA, USA, August 19–22, 2002, pp. 53-65.

[8] V. R. Basili, H. Dieter Rombach, "The TAME Project: Towards Improvement-Oriented Software Environments ", *IEEE Transactions on Software Engineering,* Vol 14(6), USA, June 1988, pp. 758-773.

[9] Fenton, N.E., and Pfleeger, S.L., *Software Metrics - A Rigorous & Practical Approach 2nd Edition*, International Thomson Publishing, Boston, MA, 1996.

[10] B. Kitchenham, S.L. Pfleeger and N. Fenton, "Towards a Framework for Software Measurement Validation", *IEEE Transactions on Software Engineering*, Vol 21(12), USA, December 1995, pp. 929-944.

[11] Distributions of derived M3 can be accessed at http://www.cs.lth.se/home/Krzysztof_Wnuk/RE_09/NumberOfChangesNeededToRemoveTheFeature.bmp

[12] D. Greer, G. Ruhe, "Software release planning: an evolutionary and iterative approach", *Information and Software Technology,* Vol 46(4), Elsevier, 2004, pp. 243-253.

[13] J. Savolainen, M. Kauppinen and T. Mannisto, "Identifying key requirements for a new product line", *Proceedings - 14th Asia-Pacific Software Engineering Conference APSEC 2007,* IEEE Computer Society, Nagoya, Japan, 2007, pp. 478-485.

[14] C. Ebert, J. De Man, "Requirements Uncertainty: Influencing Factors and Concrete Improvements", *Proceedings - 27th International Conference on Software Engineering, ICSE 2005,* IEEE Computer Society, Saint Louis, MO, United States, 2005, pp. 553-560.

[15] G. Botterweck, S. Thiel, D. Nestor, S. bin Abid, C. Cawley, "Visual Tool Support for Configuring and Understanding Software Product Lines", *Proceedings - 12th International Software Product Line Conference, SPLC 2008,* IEEE Computer Society, Limerick, Ireland, 2008, pp. 77-86.

[16] Wohlin C., Runeson P., Host M., Ohlsson M.C., Regnell and A. Wesslen, "Experimentation in Software Egnineering An Introduction", *Kluwer Academic Publishers,* USA, 2000.

[17] L. Karlsson, B. Regnell, T. Thelin, "Case Studies in Process Improvement through Retrospective Analysis of Release Planning Decisions", *International Journal of Software Engineering and Knowledge Engineering*, Vol 16(6), 2006, pp. 885-915.