Feature Transition Charts for Visualization of Cross-Project Scope Evolution in Large-Scale Requirements Engineering for Product Lines

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

In large-scale multi-project software engineering it is a challenge to provide a comprehensive overview of the complexity and dynamics of the requirements engineering process. This paper presents a visualization technique called Feature Transition Charts (FTC) that gives an overview of scoping decisions involving changes across multiple projects based on previous work on within-project visualization of feature survival. FTC is initially validated using industrial data from the embedded systems domain in a multi-project product line engineering context in dialogue with practitioners. The initial validation provided specific improvement proposals for further work and indicated a positive view on the general feasibility and usefulness of FTC.

1. Introduction

Requirements for software intense embedded systems can often be counted in thousands and often describe cutting edge functionality which can have many complex dependencies with other parts of the system. In this case, the decision about which candidate requirements should be included into the scope of the project and which should not is not always obvious. It is strongly emphasized by researchers, like for example Boehm et al. [1] and Boehm and Huang [2], that the inclusion process should be value-based, while others, for instance Wohlin et al., argue that a good understanding of the underlying decision-making process is needed so that researchers can support it in the best possible way [3].

In product line engineering, the selection process is often called *scoping* and is considered crucial for achieving economic benefits [31]. Furthermore, many embedded systems companies are releasing their products to an open market in a mode often called Market-Driven Requirements Engineering (MDRE) [4]. In this case, the complexity and uncertainty of scoping decisions may increase even more and may result in a situation where the decisions have to be made *a priori* with limited knowledge about their market value and their cost for implementation. Decisions often need to be revised due to changes in the market situation [5] or other unplanned constraints. Therefore, some requirements are deferred to later releases for a number of reasons [18]. In de-scoping there can be both rejected requirements (that for example are out of scope of the current strategy) and postponed requirements (that for example are delayed because of lack of resources). Large scope changes with many deferred requirements may significantly delay a project's overall business value, and are thus interesting to track in project and product management.

In one of our previous papers, [18], we have analyzed three large platform projects to test the applicability of our visualization technique denoted *Feature Survival Charts* (FSC) on empirical data. In the case of the company under study, the decision process is based on bundles of requirements, called *features*, rather than single requirements. A single scoping decision may concern one or many features and their dependencies. In our earlier work, we have experienced a very large number of features that were de-scoped from analyzed projects [16]. Due to one of the limitations in our previous work, namely the fact that the Feature Survival Charts only show a single project during analysis, it is not possible to analyze if some of de-scoped features are moved to another projects. In a real product development setting, we can assume that many scope changes span across several projects. Scope changes that seem to result in rejected features may in fact concern postponed features that appear in later projects. Based on these observations, the presented work addresses the following two research questions (Q2 is a refinement of the more general Q1):

Q1: How can scope changes across projects be visualized?

Q2: Which visualization mechanisms are effective in providing an overview of the timing and magnitude of feature transitions across projects in a large-scale setting?

The main contribution presented in this paper case study is a prototype visualization technique called *Feature Transition Charts (FTC)* that can show features transitions across projects, while scaling to hundreds of features. FTC has been initially validated using real data from a large-scale product line engineering case in the domain of embedded systems, and iteratively refined in dialogue with domain practitioners.

The paper is structured as follows: Section 2 provides related work. Section 3 describes background information about the context of our industrial case study. Section 4 describes the methodology used in this study. Section 5 explains our visualization technique. Section 6 describes the results from applying our technique to two industrial projects. Section 7 defines and evaluates the results. Section 8 provides conclusions and discusses their limitations.

2. Related Work

Current state-of-the-art research in software engineering has established an opinion that decision processes are the driving forces to organize a corporation's success [6]. Researchers have already contributed in creating better support for decision making based on best knowledge and experience, computational and human intelligence, as well as a suite of sound and appropriate methods and techniques [7]. The decision-making process has also been addressed by researchers working in the requirements engineering field, since it is dependent on requirements being captured, analyzed and specified before any decision about implementation can be made. The contributions in this area are visible within different aspects of requirements management, namely prioritization [8,9] or understanding of requirements dependencies [4,10]. Others have worked on connecting requirements engineering processes to decision making [11,12,13]. Finally some effort has already been dedicated to the understanding of release planning [14], while others proposed the usage of generic algorithms to plan for different releases [15].

The reasons behind scope changes have been discussed in [3,16]. Others have investigated the root cause for changing requirements, namely requirements uncertainty [17]. Selecting the appropriate set of requirements has also been addressed by researchers related to the product line community. However, the main research stream is, according to Schmid [19], focused on the identification aspect of scoping [20,21] and does not address changes beyond formal decision to approve the scope of the project.

In the requirements visualization field, the research effort is focused mainly on three aspects of requirements engineering [22]. The first aspect is addressing the problem of creating a visual representation of requirements and their attributes based on a formal language [23,24] or even visualizing these representations [25]. The second aspect addressed in the requirements visualization literature is focusing on visualization of the structure and relationships between requirements [26,27,28]. Finally, the third aspect, which is most relevant to the work presented in this paper, is addressing elicitation [29] and decision-making activities [30].

Thus the work has been conducted on release planning itself, and scoping as its vital part. However, to our best knowledge no studies have actually looked into the phenomenon of postponing features for the next release, and no studies have made an attempt to create a visual support for this aspect of product development that may help to assess its scale in real projects.

3. The case of the company under study

Our results are based on empirical data from industrial projects at a large company that is using a product line approach. 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 several 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 enables flexibility and possibilities for adaptation of the platform project, while the major release is dedicated to address functionality of the highest importance.

The company uses a stage-gate model with several increments [32]. There are *Milestones* (MS) and *Tollgates* (TG) to control the project progress. In particular, there are four milestones for the requirements management and design before the implementation starts: MS1-MS4. The scope of the project is constantly changing during this process. In this case, the project management makes scoping decisions based on groups of requirements that constitute new functionality enhancements to the platform, called *features.* The scope of each project is maintained in a document called the *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.

According to the company guidelines, most of the scoping work should be finished before reaching the second milestone of the process. After this milestone, the content of the main release of the platform project should be well defined and remain stable so that more effort can be addressed towards the preparation for the implementation phase. Therefore, minor releases are introduced to enable necessary adaptations that related product projects require. The product projects start approximately at MS2.

After MS4, the project starts its implementation phase. Even though the scope of the platform projects together with their minor releases should be defined and approved at this stage, some important changes may still happen and decisions about how to address them must be made. The changes may be related to unplanned issues with the development of previously approved features or they may be requested by important customers as a result of a rapidly changing market situation. These late changes or adaptations are usually handled by adaptations of the platform required by certain platform project releases.

4. Research Methodology

At the beginning of this study, a set of research questions and assumptions was formulated by the researchers. Researchers assumed in this case that the feature transition is a phenomenon that can have a significant representation in real life projects. It was also assumed that there is an impact of these types of changes on the quality attributes of the requirements management processes and the resulting products that should not be neglected in conscious product management. As a result, three types of transitions were defined as the most important and they are discussed in section 5. As a next step, the empirical investigation of previously derived assumptions in the given company context. In this step, we have analyzed two large platform projects. The projects under consideration contained hundreds of features, and they were related in such as way that the first one was a direct ancestor of the following one. On a set of two large projects, a name matching algorithm, checking for multiple occurrences of the same feature id among the analyzed projects, was applied to find possible reoccurrences of the same features between the projects. The result is visualized in Figure 1, a distinction between forward and backward transitions has been made, and it (the distinction) is followed by a description of the transitions in section 6.1. In the next step, each single project was analyzed for possible internal transitions. Many transitions were discovered and they are visualized in Figure 2 and 3 followed by analysis in section 6.2. Finally a multiple-step feature transition graph was proposed and it is presented in section 6.3.

As the final step of the methodology, an interview study with two practitioners, namely one requirements management process manager and one requirements engineer, was performed. The interviews lasted for about one hour each and were semi-structured. Before conducting interviews, the list of questions to ask was prepared based on the initial assumptions described in this section and in section 1. Researchers have evaluated their pre-understanding of the feature transitions phenomenon, and feedback from practitioners in the form of their suggestions for improvements was thus collected. Some of the important aspects of the discussion were the usefulness of the visualization technique presented and the importance of the need to quickly spot feature transitions in the case of the company under study. The results from this step are presented in Section 7.

5. Feature Transition Types

In this section, different types of transitions are discussed. For each platform project, there is one release, called *the major* release that provides the main part of the functionality, while other minor releases focus on adaptations and additional functionalities needed for certain products associated with them. In this context, the distinction can be made between *within-project, cross-project and multiple transitions.* Each type of transition is defined and described respectively in the following sections.

5.1 Cross-project Feature Transitions

A cross-project transition occurs when a feature is moved between two platform projects in one step. In case that a feature is moved to the following platform project, it may be included into the earliest possible release of the next platform project (the main release). However, the destination release may not always be the main release. There may also be a situation where one feature first gets internally moved to another release of its original platform project and then later moved to another platform project. This type of transition is defined in section 5.3.

There may be various reasons that cause cross-project transitions. Firstly, features may simply be moved to the next platform project due to resource constraints, secondly due to a lack of proper hardware. Thirdly because of the unfinished functionality it is difficult to minimize all non-functional issues so features may be rescheduled for a later project where they can possibly be mitigated. The decision to perform a cross-project transition should be made after a careful analysis of the impact of the transition on the included features' market-values and possible efforts for implementation. Cross-project decisions also require impact analyses to ensure that for example other features that enable new functionality to work in a new context are available. The decision should also be confirmed with a business plan for the considered functionality so that no crucial market opportunities will be missed by a decision.

5.2 Within-project Feature Transitions

This type of transition occurs between two releases within one platform project. Features are moved between releases in one step. Each platform project has in our case a set of consecutive releases that differs in providing functionality. Apart from the main release, always scheduled at the beginning of the platform project, all other releases are introduced and scheduled later in the project. Internal transitions may be caused for reasons similar to those for the external transitions: lack of resources, dependencies on suppliers or other constraints. The basic difference is that a feature internally moved is staying in the scope of its platform project while being rescheduled to a usually later release. From a business value perspective, we believe that this type of transition can be considered as less critical and to some extent positive since it enables a quicker and more flexible response to rapidly changing market situations or unplanned project difficulties.

5.3 Multi-step Feature Transitions

The last type of transition may happen both between platform project releases and the platform projects. The main difference between the previously described types of transitions and this type is that a transition is made multiple times either within one project or between different projects. The situation where a feature is moved multiple times only between the platform project's releases or between platform projects can also be classified as a multi-step transition, but we assume that it may be rare in industrial projects. Multi-step transitions can significantly influence the market-value of moved features and their cost of implementation. The management of a project can benefit from careful analysis of this type of transitions and tries to assess the impact of the transition on involved features' market value and, if applicable, their implementation cost. This type of transitions is visualized in section 6.3.

6. Visualizing Feature Transition on the Industrial Example

In order to confirm or reject our pre-understanding about described types of feature transitions, we applied a new visualization technique to data from an empirical set of two large platform projects. The characteristics of analyzed projects are presented in Table 1. The projects differ significantly in the number of features ever considered in their scope, but have a similar number of associated technical areas.

Table 1. Characteristics of analyzed projects.

Project	Nbr. of features	Percentage of internal feature transitions	Percentage of external feature transitions
A	206	17%	8%
B	568	6%	0,5%

An initial analysis of transitions present revealed that internal transitions represent 17% of all scope changes for project A, and 6% of all scope changes for project B. On the other hand, external transitions turned out to be 8% of all scope changes for project A and only 0,5% of all scope changes for project B. The numbers presented are, however, influenced by the fact that only two projects were analyzed. In general, each project will have two, or even more, associated projects; one from which the project is receiving backward transitions from and one or more to which forward transitions are sent to.

All types of transitions are visualized using a modified concept of *Feature Survival Chart (FSC)* presented in [18], namely Feature Transition Chart (FTC). The FSC, shows scope changes over time, which is illustrated on the X-axis. Each feature is positioned at a specific place along 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 in color. Based on discussions with practitioners, we decided to use the following coloring scheme: green for features considered as a part of the primary flow, red for features considered as de-scoped and, if applicable, orange, yellow, pink and cyan for other 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.

The FTC is complementing the original FSCs by marking transitions of features together with their departure and arrival points. In order to find external transitions, we have searched feature identifiers involved in both projects for exactly matching names. This technique resulted in a significant fraction of features transmitted between the projects. In order to indicate the transitions' departure and arrival points, a set of the following symbols is used. The departure points are marked by 45◦ lines leaning down if the transition is forward, or leaning up if the transition is backward. The destination points are marked by a rectangle. This technique enables the representation of the magnitude of concurrent changes in analyzed projects, which pure lines can not adduce. It may however be inefficient when many changes happen at the same time due to the overlap of symbols. Various releases within the analyzed projects are represented by various colors. Features removed from the scope of the release that they finally arrived at, or even belonged to, are marked red.

6.1. Cross-projects Feature Transitions

We have found 21 forward transitions (from Project A to Project B) in the analyzed dataset and only 4 backwards transitions (from Project B to project A). The results are depicted in Figure 1. The backward transitions are interesting to analyze since they mean that features were moved to an earlier platform project. The lines depicting transitions are not always orthogonal, which means that there has been a delay in transitions.

In order to analyze the reasons for external feature transitions, we have checked the decision logs for both projects for the descriptions of proposed changes. The

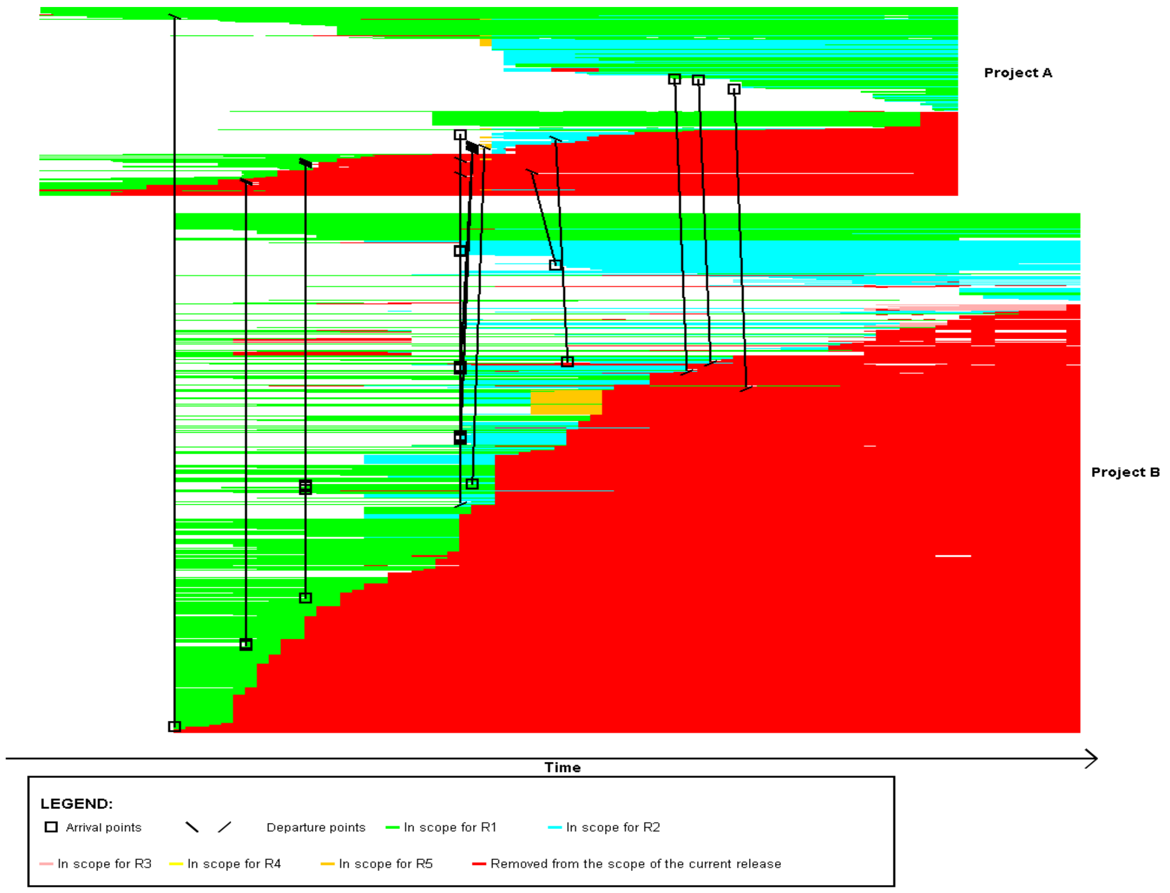

Figure 1. Cross-project transitions between the project A and B (see section 5.1). The full-size color figure can be found at http://www.cs.lth.se/home/Krzysztof_Wnuk/REV09/Figure1.bmp

analysis of forward transitions revealed that seven transitions were caused by stakeholders business decisions. The decision in these cases was to refine the features and accept only a limited scope in the next project. In three other cases, lack of development resources caused the features to be moved to the next project. On the contrary, in two other cases the resources were available, but the time schedule was too tight to be ready with the implementation of given features. In two other cases, dependencies on either suppliers or other features caused the external transitions. In one case, the feature failed compliance testing with a certain standard required by the customer so it had to be moved to the next release of the project for improvements. Finally, in two cases features were only partly ready for the original project deadline and therefore were moved to the next release. The interesting information here is that most of the functionality was available, but the company decided to postpone the commercial availability of features until the complete feature implementations were ready. The analysis of backward transitions revealed that for all cases there was a request to provide the functionality in an earlier project. The requests were accepted after checking that the development teams were capable of meeting the new deadlines and that the new features were technically compatible with the destination project's source code.

6.2 Within-projects Feature Transitions

Next we have visualized internal transitions within both projects A and B. The results are depicted in Figures 2 and 3. Various platform project releases are placed next to each other in the graphs. The time offset is not present in this case, so that all transitions are represented by orthogonal lines and transition symbols similarly to across-project transition visualization. Due to the doubling of data points (only for the features that have been moved within-project) in this type of graph, the data has been minimized by removing data points from after the transition for the source project and before the transition for the destination project. As a result, a more accurate picture of the size of various platform project releases can be achieved.

In the case of project A, we experienced in total 34 within-project transitions. 18 of them turn out to be originating from R1 and 15 from R2. All mentioned transitions are targeted to later releases. On the other hand, one transition is originating from R4 and is

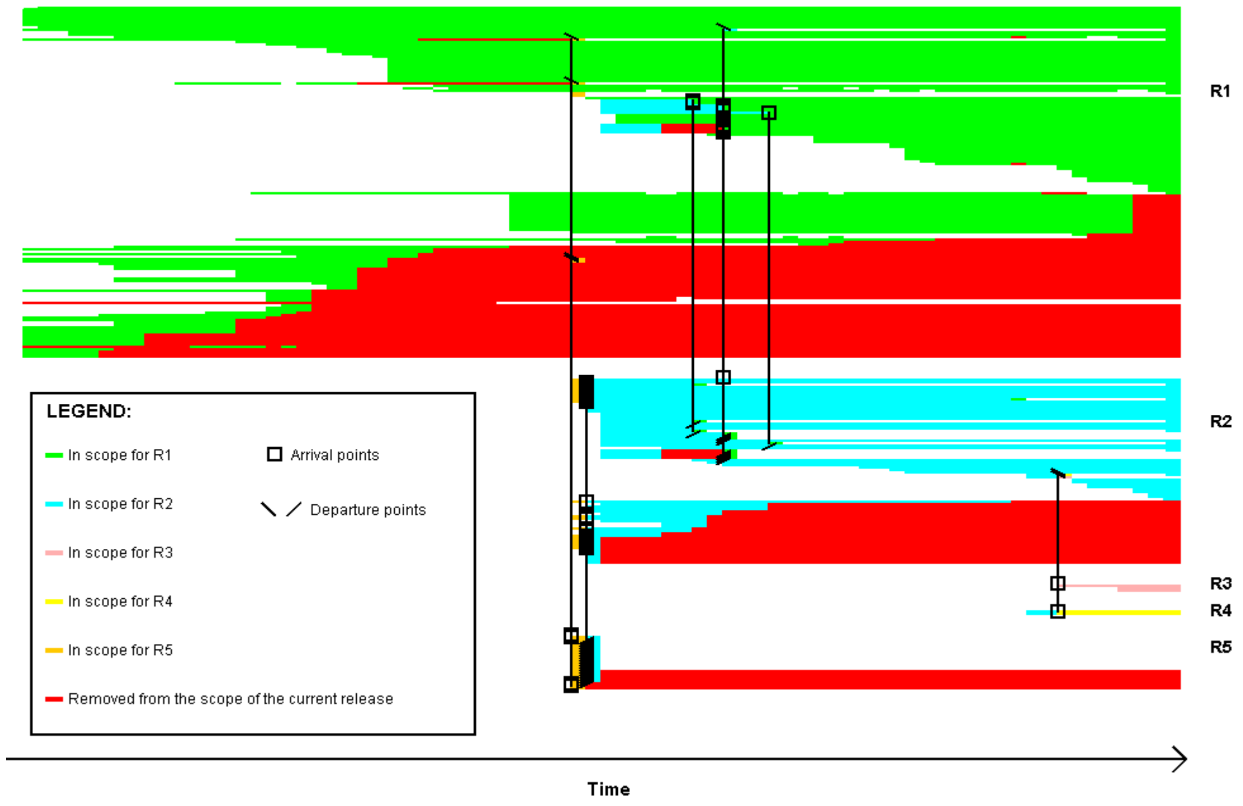

Figure 2. Within-project transitions for project A (see section 5.2). The full-size color figure can be found at http://www.cs.lth.se/home/Krzysztof_Wnuk/REV09/Figure2.bmp

directed towards an earlier scope release. In the case of project B, 36 within-project transitions were found in total. The interesting observation here is that 19 transitions are originating from release R5, another 10 from release R2. Both groups of transitions are targeted to earlier releases. In this case, only five transitions originated from release R1 and only two from release R2.

6.3 Visualizing multiple transitions.

The last type of visualization is representing only features that have been transferred multiple times. Due to the fact that these transitions are complex, the visualization used here considers only mentioned transitions. All single transitions, as well as features that were not moved anywhere, are excluded from the graph. The results from visualizing this type of transitions on the industrial data are depicted in Figure 4. In our case, only five features happened to behave in this way. For all discovered cases the scenario is the same, the features were first moved internally to an early project release within the same platform project, and then moved to the second release of the following project. To emphasize multiple transitions, a new symbol was added to the graph, namely the interim transition symbol. As a result of its design, this view cannot visualize the magnitude of multiple transitions compared with all transitions in the project. It is instead focusing on paths for multiple transitions.

7. Initial validation

As an initial validation step, interviews were conducted with two practitioners from the case of the company under study, one person working with requirements engineering process improvement and one person working with scope management. The questions were asked to confirm or reject the assumptions that the researchers had before applying visualizations to the empirical data. As one of the first questions, interest in visualizing feature transitions was discussed. Both responders expressed their interest in visualizing cross-project transitions and also reported

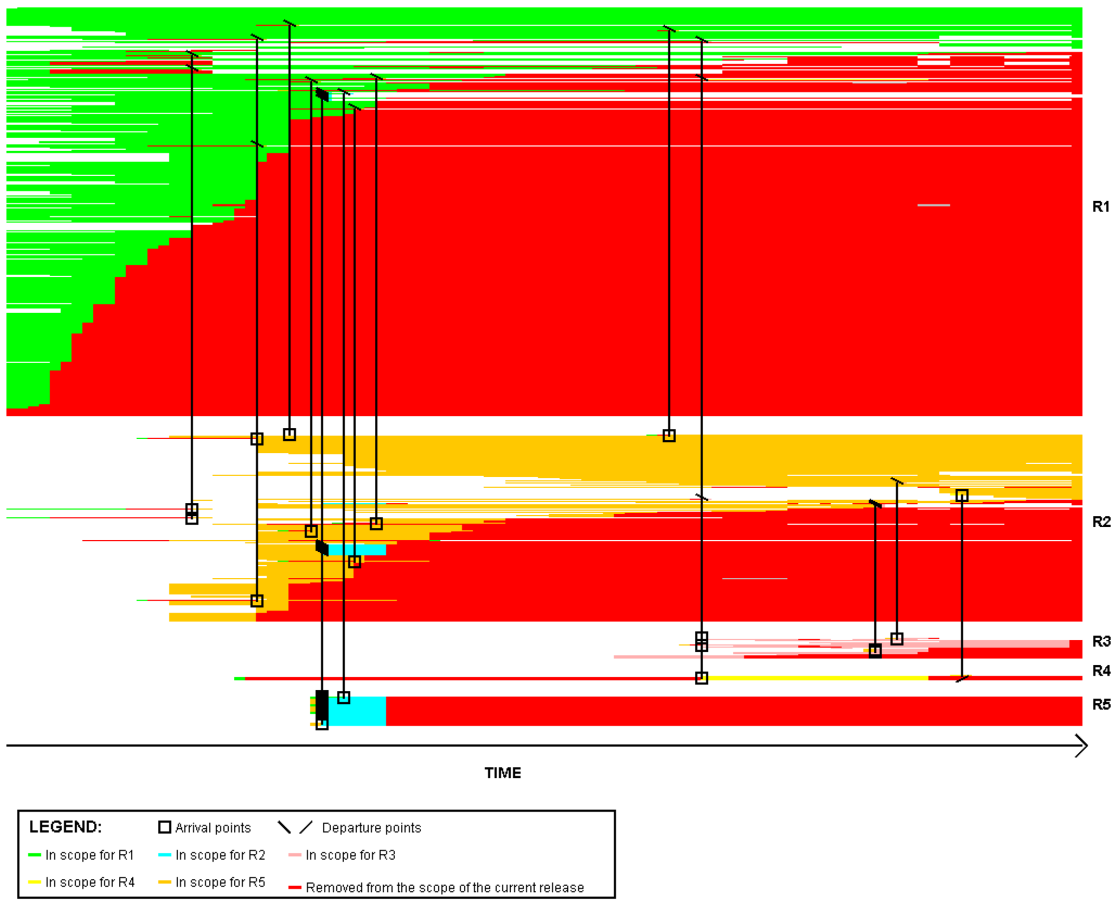

Figure 3. Visualizing within-project transitions for project B (see section 5.2). The full-size color figure can be found at http://www.cs.lth.se/home/Krzysztof_Wnuk/REV09/Figure3.bmp

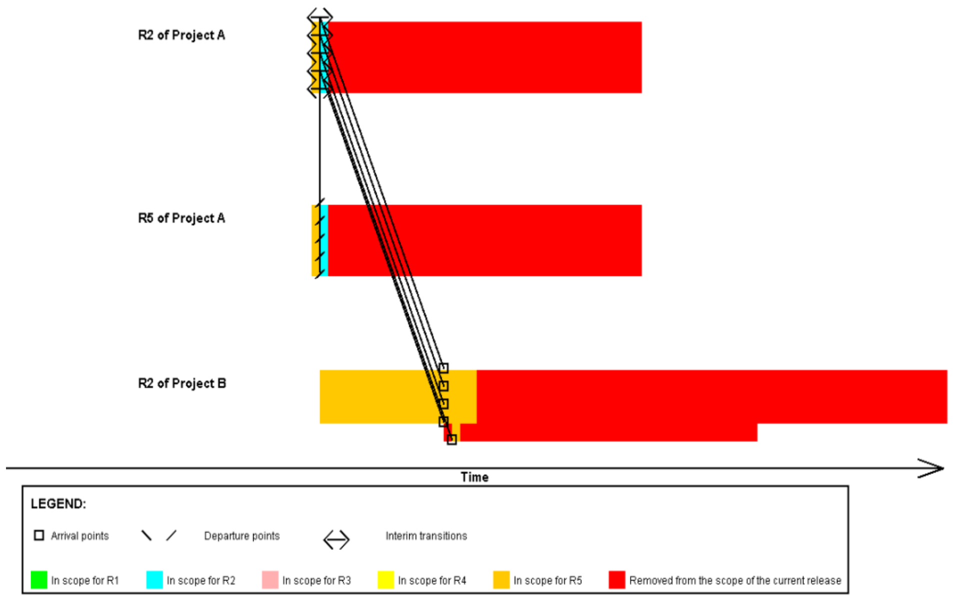

Figure 4. Multiple transitions between projects A and B visualized with the exclusion of non-transitions. The full-size color figure can be found at http://www.cs.lth.se/home/Krzysztof_Wnuk/REV09/Figure4.bmp

that current tool support cannot provide this functionality. Neither of the responders could give an accurate estimate of the scale of this phenomenon in the case of the company under study, but they agreed that there are many changes that would be valuable to visualize and analyze.

Our responders confirmed our assumptions that feature transitions may sometimes heavily influence the market value of affected features. This is because for each feature there is an optimal market window for an estimated profit. If, for some reasons, a feature is delivered to the market outside its optimal market window new estimations of its market value need to be made. In addition, cost of implementation may be affected, although market value implications were considered more important. The relation to the cost of implementation is, according to one of our responders, dependent on when the feature was moved (to which project) since that may either reduce or increase the cost for implementation. Both responders expressed that it is crucial to visualize the transitions because of so called *enablers*: features that are prerequisites of other valuable features, but that might not have a great market value on their own. Enablers often have to be implemented before, or in conjunction with, the features that rely on them. Therefore, all backward feature transitions should be analyzed to ensure that dependencies to required enablers related to moved features are available. In some cases, feature transitions may involve large architectural changes while the impact may be minimal in others. For forward transitions, enablers should not be rejected in order to make sure that support for the transferred features still persists. In the event of backwards transitions, it is important to check that support for the new functionality is available and thus may also require the backward transition of related enablers. Being able to trace features between the projects was considered as very valuable and desired.

Questions regarding usefulness and applicability of each type of the visualization were also asked. The external transitions graph was considered useful by our responders (by one responder the most useful graph). The meaning of the backward transitions was discussed together with the time delay between the exclusion and inclusion. As our responders mentioned, sometimes it is undesirable to remove the feature from the original scope until the final decision to transfer is made. For the backward transitions, the lead-time can be shown (Figure 1) representing the time needed to analyze the feature. The internal transition visualization turned out not to be as useful as the external version. Responders mentioned that the fact that each data point is placed twice on the graph (to distinguish among releases) may lead to wrong conclusions. It was also mentioned that in their company only one person is responsible for one project meaning that this person would usually be aware about the number of internal transitions in the project under his or her management.

Finally, the multiple transitions view was discussed. The responders found it less useful than the external transitions graph. One responder would like to have all features in the graph, not only the transitions, to be able to compare the scale of the project to the overall size of the project.

8. Conclusions

In this paper, we present a technique for visualization of the scope dynamics of changes within and across multiple projects called Feature Transition Charts (FTC), an extension of Feature Survival Charts (FSC) [18]. We have applied FTC post-mortem to real-world data from two large projects. FTC was initially validated in dialogue with practitioners, indicating that while FTC may be both feasible and useful, additional research could enhance the features in terms of interactive zooming and enhanced user configurability. The main findings are summarized in relation to research questions from Section 1:

- (Q1) FTC can visualize scope changes across the projects by aligning a set of FSC and depicting transfers using special markers and lines. The visualization can scale to large projects (at least in the projects we have tested), which can be counted as its advantage over a textual representation of scope dynamics The practitioners believe that FTC can give a comprehensive overview of scoping dynamics that have not previously been made explicit, and that the concept of FSC [16,18] is extended in a useful way. FTC can be used by both requirements engineers and process managers to gain valuable information about the presence and nature of scope changes across projects or projects' releases.
- (Q2) The proposed visual symbols for departure and arrivals of feature transitions can be useful in providing an effective overview of the timing and magnitude of feature transitions. However, in a very large scale projects, many adjacent transitions can overlap and future work thus may include experiments with interactive zooming and filtering features.

Limitations. Our study has some limitations. Firstly, even if the case of the company under study 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. Secondly, our initial validation of FTC is limited to a static post-mortem analysis and because of that it could not be applied in a proactive manner and no feedback from ongoing projects could be gathered. Thirdly, when the size of the projects grows, our visualization technique should be complemented by zooming and interactive features so that the holistic picture can be perceived, while the details are available on demand.

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. Other means of marking the departure and arrival points should be evaluated. Finally, additional work should be performed to address the applicability of FTC in other contexts, for example other domains, such as information systems, and other development modes, such as single product development or agile development.

Acknowledgements. This work is supported by VINNOVA (Swedish Agency for Innovation Systems) within the UPITER project. Special acknowledgements to Thomas Olsson for valuable help with section 7, and to Lars Nilsson for valuable language comments.

9. References

[1] B. W. Bohem, "Value-based software engineering", *Software Eng. Notes 28(2)*, ACM SIGSOFT, 2003, pp. 1-12.

[2] B.W. Bohem, L. G. Huang, "Value-based software engineering: A case study.", *Computer,* IEEE Computer Society, March 2003, pp. 33-41.

[3] 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, IEEE Computer Society,* Piscataway, NJ 08855-1331, United States, 2005, pp. 246-255.

[4] P. Carlshamre, B. Regnell, "Requirements lifecycle management and release planning in market-driven requirements engineering processes", Proceedings 11th International Workshop on Database and Expert Systems Applications, *IEEE Computer Society*, Los Alamitos, CA, USA, 2000, pp. 961-965.

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

[6] G. De Gregorio, "Enterprise-wide Requirements and Decision Management", Proc. 9th International Symposium of the International Council on System Engineering, Brighton, 1999, pp. 1-7.

[7] G. Ruhe, "Software Engineering Decision Support – A new Paradigm for Learning Software ", *Lecture Notes in Computer Science*, vol. 2640, Springer-Verlag, Berlin, 2003, pp. 104-113.

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

[9] J. Karlsson, C. Wohlin, B. Regnell, "An evaluation of methods for prioritizing software requirements." *Information*

and Software Technology, 39 (14-15), 1997-1998 pp. 939-947.

[10] Å. Dahlstedt, A. Persson, „Requirements interdependencies – Moulding the state of research into a research agenda." Proceedings Ninth International Workshop on Requirements Engineering (REFSQ'03), *Klagenfurt/Velden*, Austria, 2003, pp. 71-80.

[11] A. Aurum and C. Wohlin, "Applying decision-making models in requirements engineering.", Proceedings of Requirements Engineering for Software Quality, *Information and Software Technology 45(14),* Elsevier, Essen Germany, December 2002, pp. 2-13.

[12] A. Aurum, C. Wohlin, "The fundamental nature of requirements engineering activities as a decision-making process". *Information and Software Technology*, 45(14), 2003, pp. 945-954.

[13] B. Regnell, M. Host, J. Natt och Dag, and A. Hjelm, "Case study on distributed prioritization in market-driven requirements engineering for packaged software.", *Requirements Engineering 6(1)*, Springer-Verlag, London, 2001, pp. 51-62.

[14] P. Carlshamre, "Release planning in market-driven product development: Provoking an understanding", *Requirements engineering 7(3)*, Springer-Verlag, London, 2002, pp. 139-151.

[15] G. Ruhe, D. Greer, "Quantitative studies in software release planning under risk and resource constraints", Proceedings of International Symposium on Empirical Software Engineering (ISESE), IEEE, Los Alamitos CA, 2003, pp. 262-271.

[16] K. Wnuk, B. Regnell and L. Karlsson, "What Happened to Our Features? Visualization and Understanding of Scope Change Dynamics in a Large-Scale Industrial Setting", Accepted as a publication for the 17th International Requirements Engineering Conference, RE09, Atlanta, Georgia, USA, 2009.

[17] 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.

[18] 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), *IEEE,* Spain, 2008, pp.41-50.

[19] K. Schmid, "A Comprehensive Product Line Scoping Approach and Its Validation", 24th International Conference on Software Engineering (ICSE 2002*), IEEE Computer Society*, Orlando USA, May 19-25 2002, pp. 593-603.

[20] 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.

[21] 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.

[22] O. C.Z. Gotel, F. T. Marchese, S.J. Morris "On requirements visualization", Second International Workshop on Requirements Visualization (REV 2007), *IEEE Computer Society*, New Delhi India, 2007, pp. 80-9.

[23] Teyseyre, A. "A 3D Visualization Approach to Validate Requirements", Proc. Congreso Argentino de Ciencias dela Computacion, Argentina, October 2002.

[24] Unified Modeling Language webpage, www.uml.org last visited July 2008.

[25] S. Konrad, H. Goldsby, K. Lopez, B. H.C. Cheng, "Visualizing Requirements in UML Models", First International Workshop on Requirements Visualization (REV 2006), *St. Paul MN* United States, September 2006.

[26] C. Duan and J. Cleland-Huang, "Visualization and Analysis in Automated Trace Retrieval", First International Workshop on Requirements Visualization (REV 2006), St. *Paul MN* United States, September 2006.

[27] O. Ozakaya, "Representing requirements relationships", First International Workshop on Requirements Visualization (REV 2006), *St. Paul MN* United States, September 2006.

[28] D. Sellier, M. Mannion, "Visualizing Product Line Requirements Selection Decision Inter-Dependencies", Second International Workshop on Requirements Visualization (REV 2007), *IEEE*, New Delhi India, 2007, pp. 1-10.

[29] M. Pichler, H. Humetshofer, "Business Process-based Requirements Modeling and Management", First International Workshop on Requirements Visualization (REV 2006), *St. Paul MN* United States, September 2006.

[30] M. S. Feather, S. L. Cornford, J. D. Kiper, T. Menzies, "Experiences using Visualization Techniques to Present Requirements, Risks to Them, and Options for Risk Mitigation", First International Workshop on Requirements Visualization (REV 2006), *St. Paul MN* United States, September 2006.

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

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