Variability-Based Release Planning

Samuel Fricker[1] and Susanne Schumacher[2]

[1] Blekinge Institute of Technology, School of Computing
Campus Gräsvik, 371 79 Karlskrona, Sweden
samuel.fricker@bth.se
[2] Zurich University of the Arts
Ausstellungsstrasse 60, 8005 Zurich, Switzerland
susanne.schumacher@zhdk.ch

Abstract. A release plan defines the short-term evolution of a software product in terms of development project scope. In practice, release planning is often based on just fragmentarily defined requirements. Current release planning approaches, however, assume that a requirements catalogue is available in the form of a complete flat list of requirements. This very early commitment to detail reduces the flexibility of a product manager when planning product development. This paper explores how variability modeling, a software product line technique, can be used to plan, communicate, and track the evolution of a single software. Variability modeling can reduce the number of decisions required for release planning and reduce the information needed for communicating with stakeholders. An industrial case motivates and exemplifies the approach.

Keywords: release planning; variability; features.

1 Introduction

Release planning is used to plan the development of software products by allocating requirements to development projects [1]. Thereby, the evolving product is aligned with market and stakeholder needs, company objectives, and constraints such as time, resources, and legacy. Release planning is also a central concern in iterative development, where the scope of iterations, rather than projects, is defined [2].

Release planning involves the following steps [3]. Requirements are elicited and specified based on an understanding of stakeholders, organizational environment, and culture [4]. That same understanding is a basis for defining criteria [5] to evaluate and prioritize requirements [6]. The priorities are then used to scope releases by allocating requirements to development projects. The resulting release plans are implemented, delivered, and analyzed with post-release reflections [7].

Release planning is challenging in practice because of the typically large number of requirements [8]. These requirements often are not properly defined and detailed due to the limited effort that can be invested before a development project is funded. These challenges contradict with the assumptions taken by current release planning approaches. For example, requirements catalogues cannot be considered complete or correct until their effect on solution architecture and project planning has been understood and agreed [9, 10].

B. Regnell, I. van de Weerd, and O. De Troyer (Eds.): ICSOB 2011, LNBIP 80, pp. 181–186, 2011.

This paper proposes to ease the release planning problem not by improving prioritization algorithms, but by structuring the requirements catalogue. The approach is based on analyzing variability [11] by utilizing AND, OR, and REQUIRES relationships between requirements [12]. Such variability is a particular form of decision options [13] utilized to define the scope of product releases.

The approach works with a body of only fragmentarily and vaguely defined requirements and can be used in continuous agile product management processes [14]. Simple abstractions allow incremental aggregation and extension of requirement catalogues, hence address the scalability problem [15]. The abstractions further allow expressing the essence of release decisions to support the dialogue of the product manager with stakeholders, hence addressing the problem of limited trust that results from mathematical, black box-oriented prioritization [15].

This paper motivates and describes the variability-based release planning approach. An industrial case of an organization that transited from flat requirements list-based release planning to variability-based release planning is used to illustrate the approach. The paper is structured as follows. Section 2 describes background and motivation. Section 3 introduces variability-based release planning. Section 4 provides a short discussion and concludes.

2 Background and Motivation

Current release planning approaches require a flat and complete catalogue of requirements that are evaluated, prioritized, and selected for implementation [16]. Known approaches include manual techniques such as top ten, numerical assignment, raking, and 100$-test [6], and computer-based techniques such as Integer Linear Programming [17] and the Analytical Hierarchy Process [18].

We investigated release planning in a company that offered software as a service for managing media such as text, sound, pictures, and movies. The product manager and important stakeholders, henceforth called “release planners”, regularly planned software releases that corresponded to small and large version increments.

The requirements catalogue was managed in a word processor document and used as a basis for release planning. It contained 108 requirements. Some of them were specified with a few words in a declarative manner. Others were specified in detail with descriptions of up to 245 words. The requirements were grouped into 12 sections and 19 subsections or themes. The grouping, however, did not show a relationship with requirements allocation to development releases. In average, a group contained 3.6 requirements and was allocated to 1.93 releases.

Current release planning approaches would have expected the release planners to evaluate every detailed requirement. For example the planners would have compared each requirement individually with other requirements by posing questions such as “are *thumbnails of variable sizes* (requirement 11.3) more important than *storage of search results* (requirement 8.1)?” Such evaluation would have led to detailed evaluation results. However, it also would have increased the risk of losing the understanding of the big picture and sub-optimizing details irrelevant in the given product evolution stage.

Considering requirements out of their larger context also would have increased the risk of misunderstandings: *When would thumbnails be shown? For what purpose? Which sizes? What (photos, videos, documents, etc.) would be depicted by these thumbnails?* It is evident, independent of the applied prioritization technique, that the lack of a common understanding leads to considerably different interpretations of criteria such as importance, urgency, dependencies, implementation cost, and risk.

These problems motivated us to indentify alternative release planning approaches based on the following criteria. *Minimalism*: release planning should involve as few decisions as possible to reduce effort and likelihood of errors. *Traceability*: a release plan should be traceable to roadmaps to align long-term with short-term planning. *Saliency*: a release plan should abstract detail and contain just salient information to support negotiations and communication. *Evolution*: a release plan should change to reflect evolving knowledge and progress.

A solution that respects these criteria moves the release planning problem away from mathematical optimization towards supporting the dialogue with stakeholders. A minimal number of release planning decisions enables stakeholders to reflect on alternatives, for example by building a mental model of the decision options and by exploring what-if scenarios. Traceability to roadmaps allows stakeholders to understand the impact of decisions on their stakes and commitments regarding the software product. For example they need to know whether they need and can hold promises given during long-term planning. Saliency allows focusing on the big picture, e.g. by discussing just the most important decision-making topics and options during steering committee meetings and by communicating just the important information for marketing purposes. A release plan finally, is in continuous evolution. It is expanded when new requirements are elicited, refined when solutions are explored, and its status changed when development progresses.

3 Variability Modeling for Release Planning

Feature trees are a widespread approach to document and analyze variability of software products [11]. They are used to specify how features vary for the products of a product line (variability in space). Applied for release planning, variability models can be used for defining the evolution of software (variability in time) [19]. How feature trees are utilized for release planning, has not been researched yet though [1].

The here proposed approach structures requirements with such a feature tree. The tree's root refers to the central parts of the solution: the architecture and infrastructure assets to be developed before value-adding features can be added. The branches capture variability by referring to features that are enhanced incrementally with extending sub-features. Each feature groups requirements with an AND relationship [12]. These requirements are intended to be developed together in the same development increment. An enhancing feature stands in a REQUIRES relationship with the enhanced feature. Alternative enhancements stand in an OR relationship.

A product manager constructs a feature tree by first grouping requirements into coarse features and then building feature vectors [20] that connect the root of the feature tree with leafs. Feature vectors are built iteratively by extracting requirements from given features into extending sub-features [21]. The feature vector-building process stops when no requirements can be extracted without making the concerned super-feature useless.

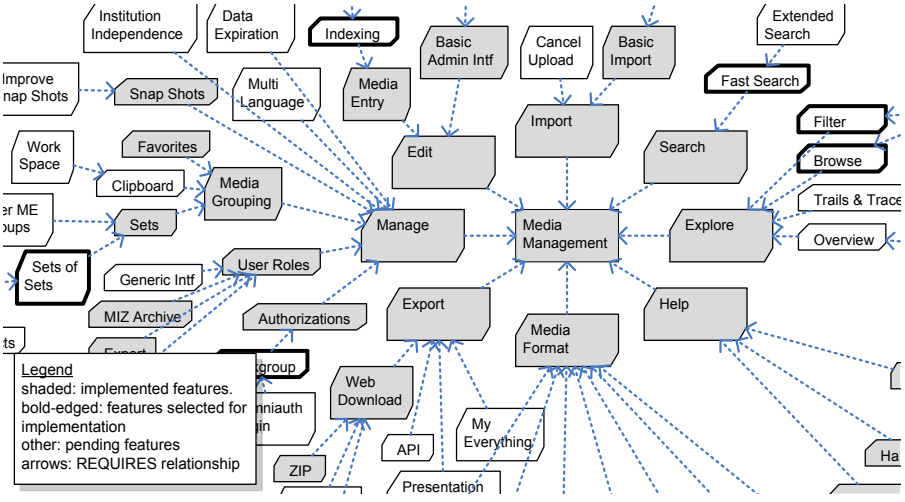

Fig. 1. Extract of the media management solution's feature tree (notation [21]). Allocation of requirements to features (AND relationship) not shown

The product manager uses the feature tree for release planning, communication, and controlling. Initial development starts with the root of the tree. Release planning involves selecting those features that are not implemented yet from those that are connected with already implemented or planned features (connectivity rule). Implementation progress is documented by tagging features as being implemented. The tree is used in project status and in steering committee meetings as an instrument to illustrate plans and progress. Emerging requirements, for example discovered during elicitation activities or late in the development process, are added to existing non-implemented features or as new leaf features to the tree.

Figure 1 shows the feature tree that the product manager of the media management solution created and continuously used for planning, communicating and controlling implementation progress. *Media Management* was the root and referred to the basic software infrastructure. Each node selected for implementation was marked with a bold edge and the contained requirements were entered into the backlog of the concerned development increment. The feature *Indexing*, for example, contained 12 requirements (AND relationship). The feature *Indexing* was only selected for implementation after *Media Entry* was implemented (REQUIRES relationship). Concluded feature implementation was marked by shading the corresponding nodes. In an earlier development stage, *Media Entry* stood in competition with *Basic Admin Intf* (OR relationship): none, one of them, or both features could be implemented as an enhancement of *Edit*. The chain *Edit* ← *Media Entry* ← *Indexing* represented one of the feature vectors of the media management solution.

A major challenge concerned the implementation of the approach. No tools were available that were integrated into the development environment. Ad-hoc tools were used instead. The feature tree was specified with diagramming software and the requirements–feature allocation (AND grouping of requirements) with a word processor document. The development project tracked requirements and implementation

progress with a task management solution. The product manager ensured consistency in meetings with the development team.

4 Discussion and Conclusions

This paper has introduced an approach, variability-based release planning, that simplifies release planning by structuring the underlying requirements catalogue. It utilizes feature trees and feature vectors to structure requirements with AND, OR, and REQUIRES relationships. It uses feature extraction as the basic technique to construct the tree and a simple lifecycle model to track and plan software development. An industrial case was described to motivate and exemplify the approach.

Variability-based release planning reduces the complexity of release planning compared with the traditional approaches that are based on a flat list of requirements. The feature tree allows abstracting from detailed requirements to groups of requirements. It provides a graphical representation that allows stakeholders to develop a mental model of the decision options. A planned release can now be described by referring to a few features instead of a potentially large and incomplete set of requirements. The described tree construction, planning, and controlling approach is used to evolve the information base for release planning. Containers for adding requirements that are discovered late in the development process allow dealing with requirements incompleteness. The connectivity rule provides support to rapidly identify those features that are candidates for release planning. Release planning scenarios can be analyzed by exploring the OR relationships between sub-features. Roadmaps can be traced to by referring with important features to agreed activities of the company, for example by utilizing appropriate naming.

Variability-based release planning has not been integrated with other product management and development activities yet. Future research should investigate how the approach relates to upstream techniques for requirements triage such as the requirements abstraction model [4] and downstream techniques for requirements communication such as handshaking with implementation proposals [9]. Empirical research is needed to better understand benefits, and limitations of variability-based release planning in practice, for example compared to other release planning approaches.

References

1. Svahnberg, M., Gorschek, T., Feldt, R., Torkar, R., Bin Saleem, S., Usman Shafique, M.: A Systematic Review on Strategic Release Planning Models. Information and Software Technology 52, 237–248 (2009)
2. Cohn, M.: Agile Estimating and Planning. Prentice-Hall, Englewood Cliffs (2006)
3. Amandeep, N.F.N.G., Ruhe, G., Stanford, M.: Intelligent Support for Software Release Planning. In: Bomarius, F., Iida, H. (eds.) PROFES 2004. LNCS, vol. 3009, pp. 248–262. Springer, Heidelberg (2004)
4. Gorschek, T., Wohlin, C.: Requirements Abstraction Model. Requirements Engineering 11(1), 79–101 (2006)
5. Wohlin, C., Aurum, A.: What is Important when Deciding to Include a Sotware Requirement into a Project or Release. In: International Symposium on Empiricial Software Engineering (2005)

6. Berander, P., Andrews, A.: Requirements Prioritization. In: Aurum, A., Wohlin, C. (eds.) Engineering and Managing Software Requirements. Springer, Heidelberg (2005)
7. Karlsson, L., Regnell, B., Karlsson, J., Olsson, S.: Post-Release Analysis of Requirements Selection Quality - An Industrial Case Study. In: 9th International Workshop on Requirements Engineering: Foundation for Software Quality, RefsQ 2003 (2003)
8. Regnell, B., Svensson, R.B., Wnuk, K.: Can we beat the complexity of very large-scale requirements engineering? In: Rolland, C. (ed.) REFSQ 2008. LNCS, vol. 5025, pp. 123–128. Springer, Heidelberg (2008)
9. Fricker, S., Gorschek, T., Byman, C., Schmidle, A.: Handshaking with Implementation Proposals: Negotiating Requirements Understanding. IEEE Software 27(2), 72–80 (2010)
10. Fricker, S., Glinz, M.: Comparison of Requirements Hand-Off, Analysis, and Negotiation: Case Study. In: 18th IEEE International Requirements Engineering Conference, Sydney, Australia (2010)
11. Schobbens, P.-Y., Heymans, P., Trigaux, J.-C., Bontemps, Y.: Generic Semantics of Feature Diagrams. Computer Networks 51, 456–479 (2007)
12. Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B.: Nattoch Dag, J.: An Industrial Survey of Requirements Interdependencies in Software Product Release Planning. In: 5th IEEE International Symposium on Requirements Engineering (2001)
13. Haberfellner, R., Nagel, P., Becker, M., Büchel, A., von Massow, H.: Systems Engineering: Methodik und Praxis, 11th edn. Verlag Industrielle Organisation (2002)
14. Vlaanderen, K., Jansen, S., Brinkkemper, S., Jaspers, E.: The Agile Requirements Refinery: Applying SCRUM Principles to Software Product Management. In: 3rd International Workshop on Software Product Management (2009)
15. Lehtola, L., Kauppinen, M.: Suitability of Requirements Prioritization Methods for Market-driven Software Product Development. Software Process Improvement and Practice 11, 7–19 (2006)
16. Carlshamre, P.: Release Planning in Market-Driven Software Product Development: Provoking an Understanding. Requirements Engineering 7, 139–151 (2002)
17. Ruhe, G., Saliu, M.O.: The Art and Science of Software Release Planning. IEEE Software 22(6), 47–53 (2005)
18. Karlsson, J., Ryan, K.: A Cost-Value Approach for Prioritizing Requirements. IEEE Software 14(5), 67–74 (1997)
19. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering: Foundations, Principles and Techniques, 1st edn. Springer, Heidelberg (2005)
20. Nejmeh, B., Thomas, I.: Business-Driven Product Planning Using Feature Vectors and Increments. IEEE Software 19(6), 34–42 (2002)
21. Stoiber, R., Glinz, M.: Feature Unweaving: Efficient Variability Extraction and Specification for Emerging Software Product Lines. In: 4th International Workshop on Software Product Management (IWSPM 2010), Sydney, Australia (2010)