

Using Portfolio Theory to Support Requirements Selection Decisions

Nina D. Fogelström, Emil Numminen, Sebastian Barney

Blekinge Institute of Technology

Karlskrona, Sweden

nino.dzamashvili.fogelstrom@bth.se, emil.numminen@bth.se, sebastian.barney@bth.se

Abstract—Selecting requirements for a release of software is a difficult undertaking as people have trouble comparing requirements of different types and have natural biases towards short-term gains over longer-term sustainability. Portfolio theory is proposed as a solution to this problem, as it provides a method for balancing investment options to maximize the likelihood of a given return. This approach is explored generally and through an example. The results suggest portfolio theory can be applied for this purpose. Applying portfolio theory to determine the amount of development time that should be spent on different types of requirements shows the most potential, especially when data on expected risks and returns is limited.

Market-driven development, software product, requirements selection, product management, portfolio theory, real options theory.

I. INTRODUCTION

Software is increasingly being developed for the mass-market rather than for a specific customer [1]. This type of development brings specific challenges to development organizations, with one of the main challenges being to decide which functionality should go into each release of the product [2]. Planning functionality over product releases is tightly connected to software product management and requirements selection decision-making. These decisions should ensure that both short and long-term business goals defined in the company and product strategy are realized [3] [4].

Requirements selection decisions are considered to be complex, since decision makers have to choose between large numbers of requirements, representing interests of different stakeholders [2]. Further, these decisions are often made under uncertainty, where the value and ROI of requirements is not always clear [5]. For example, in order to achieve sustainable growth and success, software companies need to not only satisfy current market needs, which usually are associated with short-term benefits and low investment risk, but also invest in innovations and product architecture, which are associated with long-term benefits and higher investment risk [6, 7]. Recently conducted industrial studies in this area show that finding a balance between investments in requirements representing current market needs, innovations, quality aspects and product architecture improvements is difficult [8, 9].

This paper examines opportunities provided by portfolio theory [10] to handle the uncertainty in requirements selection decisions and achieve a better balance between

different requirement types, such as current market needs, innovations and investments in software product architecture.

The paper is outlined as follows: Section II presents requirement types involved in requirements selection decisions and a concrete example of a requirements selection decision problem. Section III introduces portfolio theory and discusses its usage in the given context. Section IV provides discussion on the suggested approach, while our conclusions and plans for future work are presented in Section V.

II. BACKGROUND

When planning future releases of a software product, a decision must be made as to which of the large number of potential requirements will be implemented, and in which release each will be assigned [2]. The potential requirements can be classified as follows: *Customer specific*, *Market-pull*, *Innovation*, *External quality* and *Internal quality*. Recent industrial studies have shown that these requirement types are associated with different level of investment risk [9, 8, 11].

Customer specific requirements represent the requirements of some key customer(s). These are associated with low level of investment risk, since the customer value of a requirement is known. *Market-pull requirements* are also associated with lower risk levels, as they represent current market trends, which are somewhat predictable. *Innovation requirements* represent features or functionality that is new and not really represented by the current market demands. Therefore they are associated with higher levels of investment risk [9].

External quality requirements represent quality attributes that are visible to the customer, such as performance, reliability, and usability. *Internal quality requirements* are quality attributes representing software product architecture, such as updateability, modularity, testability and maintainability. Compared to *External quality requirements*, investments in *Internal quality requirements* are associated with higher levels of risk since *Internal quality requirements* are not directly visible to the customer and thus do not have a direct link to customer value [9].

Decisions on what requirements are included in a product are based on following criteria¹: *Requirements value*, (represented in terms of its contribution to the product sales [12, 13], or requirements contribution to the decreased

¹ Additional criteria also exist, but cost and value are the base criteria used different release planning models.

product development costs [11]) and *Requirement cost* (costs required for developing a specific requirement).

Table 1 models an example of requirements selection decision problem involving the requirement types presented above. The *cost* column estimates total cost of the requirements in each category. The other columns represent requirements expected contribution either to product sales or in decreasing of overall product development costs. The numbers presented in the rows are fictitious, however, they are designed to represent the investment risk and level of expected contribution of each requirement type.

Table 1: Requirements selection decision problem

Req. Type	Cost	2010-2011		2012-2014	
		Contrib. increased product sales (%)	Contrib. decreasing overall develpmt costs (%)	Contrib. increased product sales (%)	Contrib. decreasing overall develpmt costs (%)
Customer specific	100	5 - 7	0	0	0
Market pull	200	20 - 30	0	5 - 10	0
Innovation	300	0 - 50	0	10 - 30	0
External quality	250	5 - 20	0	0	0
Internal quality	400	0	0 - 5	0	15 - 30

For example, the *customer specific requirement* is expected to contribute to sales between five and seven percent. This contribution is relatively small compared to the *innovation*, which is expected to contribute up to 50% to the expected product sales. However, the investment risk for the *customer specific requirement* is considerably lower than the *innovation*, as the variability of increased sales for the *innovation* is much larger than the variability on increased sales for the *customer specific requirement*.

Existing research [8, 9] has shown that due to lower investment risks with customer specific and market-pull requirements, these are consistently prioritized over innovations and quality attributes. It is also worth noting the situation with internal quality requirements. As shown in Table 1 these requirements are not directly associated to increased sales of a product, but rather contribute by decreasing the overall development costs. Further, investments in these requirements are often long-term. This also contributes that investments in internal qualities are often down prioritized, creating a problem in the long-term sustainability of a product and company's business [6].

The next section uses the example provided above to show how an application of portfolio theory helps find a better balance between different types of requirements.

III. OPPORTUNITIES PROVIDED BY PORTFOLIO THEORY

Portfolio theory is a branch of financial economics that aims to managing the risk of an investment strategy for a given expected return [10, 14, 15]. The risk of a portfolio is measured as the variance of the expected return of the portfolio. The two main variables for doing this is by choosing in which assets an investment is made (i.e. software requirements in this paper) and what weights the different investments get from the budget total constraint. Depending

on the return characteristics of the different assets, there exist some combinations and weights of assets that result in lower risk than the alternatives. This can be seen if the variance of the portfolio is graphed against the expected return of the portfolio, see e.g. Numminen [16]. The variance of a portfolio is calculated as:

$$\sigma^2 = \sum_{i=1}^n \sum_{j=1}^n w_i w_j \rho_{i,j} \sigma_i \sigma_j$$

where w denotes the weight of the investment in the asset, ρ denotes the correlation between the expected return of the assets and σ denotes the standard deviation of the of the expected return of the asset i.e. the square root of the variance of the expected return. To calculate the weights of the minimum variance portfolio, one sets-up the Lagrange function for the unique variance function and solves for the first-order condition for which the unique weights of the different assets can be solved.

The graph plots the relationship between the variance and expected return of the portfolio in the shape of a convex hyperbola, since the variance function is a C^2 -function. The portfolios of requirements that lie on the line are the portfolios that the rational software developer will invest in, and it is these combinations that are called efficient frontier of portfolios. Given the characteristics of the frontier of efficient portfolios there exists two or more combinations of investments in set of assets that results in the same risk. The difference between these portfolios is that one of them will have a higher expected return for the same development risk. All efficient portfolios stems from the minimum-variance portfolio, which is what a risk-averse firm will invest according to when developing software. The efficient portfolios are the ones that lie on the northeast of the minimum-variance portfolio along the efficient portfolio frontier (see Numminen [16]).

In the context of requirements selection decision problem presented in Section II, portfolio theory presents an opportunity to encourage investments in requirement types such as innovation and internal quality, by providing a way to manage investment risk associated with these requirement types. More specifically, given the economic characteristics of different types of requirements presented in Table 1, portfolio theory can be used to find different target weights in the different types of requirements in order to lower the short-term and long-term risk of software development (see Numminen [16]). This may lead to some requirements being given a higher prioritization in the short-term, although that they may not demanded as highly as other requirements (e.g. investments in internal and external quality improvements).

Depending on the return or risk characteristics that the developing firm wants on its developments, the firm can use the portfolio approach to prioritize among its investments in the different requirements. If the firm wants to minimize the risk in the development it should invest according to the weights that minimize the variance in the above formula. If the firm wants to increase the expected return in the project the firm should increase the investment according to the weight that constitutes the efficient frontier based on the

minimum variance portfolio. On a general level, portfolio theory constitutes a way for the firm to prioritize between different requirements based on the risk and expected return characteristics that the firm wants on the software developing projects. By using this approach the firm also controls the two main variables in the development from an economic point of view.

While this approach reduces the overall long-term development risk for the firm it also creates opportunities for the firm for future developments of applications based on these investments. Risk reduction by applying portfolio theory thus creates real options for the firm. A real option can be defined as the right, but not the obligation, to undertake an action for the future at a predetermined cost for a predetermined period of time [17]. Other researchers use similar definitions for real options [18, 19]. Having these options gives the firm a proactive approach to development, where the firm can react faster on to changes in customer demands compared with if the firm only focuses on what the customers' demand in the short-term. Thus this decreases the volatility in the sales for the firm, if it can capitalize on first-mover advantages [20] in the market. This enables the firm to have a robust adaptive development strategy [21], which will be more fit under different market conditions over time. So requirements engineering according to portfolio theory does not only reduce the risk in software development but also decreases the volatility in the revenues for the firm. It must however be noted that all opportunities created cannot be capitalized and it is only when the created options are exercised that monetary value is created for the firm. But, all future opportunities that are created do create adaptability for the firm under competitive market conditions and hence create protections for bad outcomes or possibilities for new business opportunities.

The approach discussed above has been criticized for a mismatch between the modern portfolio selection problem and requirements engineering in software development [22]. In this paper, portfolio theory is not used for balancing the overall risk between current software projects and previous software projects, which is why the criticism does not apply. Instead we view portfolio theory as an *a priori* tool for the firm to use to manage the investments that the firm plans in different requirements. By using portfolio theory in this way the mismatch discussed by Verhoef [22] is no longer valid.

IV. DISCUSSION

Prior research has used portfolio theory to better understand the value of software requirements [13]. The approach proposed in this paper takes this work one-step further by proposing an approach to assist in the prioritization and selection of requirements for a release of software. Portfolio theory assists decision makers balance short and long-term goals, unlike AHP and multi-criteria decision-making.

Portfolio theory can be applied to requirements selection decisions in two different ways. It can be applied directly to a set of requirements; however, this assumes that cost, as well as short and long-term impact on sales and reduced development cost is known for each requirement. Few

software development organizations are likely to be sufficiently mature to realize such an approach.

A more general approach would be using portfolio theory on requirement types instead of specific requirements. In this case each requirement type generally has a similar profile, for example see Table 1. It is also easier to collect and calculate information about a type of requirement than a specific requirement. This allows portfolio theory to be applied directly to the requirement type categories. In this case the resultant set of numbers indicate the relative investment that should be made to each requirement type. For example, the result could suggest that optimally 20% of the development time should be spent on *internal quality requirements*.

The ability to calculate the time to be spent on each requirement type is an important result. While portfolio theory can be used to calculate these times, other methods can be used to prioritize requirements within each set and determine what exactly is done with that time. This addresses the problems previously presented, where people have trouble prioritizing requirements of different types [8], but find prioritizing requirements of a single type a much easier task.

Further, this approach removes the need for a single person to be responsible for all requirement prioritization and selection decisions – which requires a broad and deep understanding of the entire product. As the time allocated to each set of requirements is fixed for a release of the software, it is possible to get different people to prioritize the requirements in each group. This allows the most informed people to make decisions about each requirement type. Agile literature, for example, has found that developers are better at prioritizing and selecting internal quality requirements than other groups as these decisions will most directly affect them [23].

However, caution must be taken with a distributed approach to requirements selection. The product manager must ensure that the balance between the different requirement types remains appropriate for each release. It is likely the balance will change with time and product maturity. Effort also needs to be made that all of the decision makers have a common understanding of the product and company roadmaps and strategies, as these will influence the decisions made by each of the people involved.

V. CONCLUSIONS AND FUTURE WORK

Requirement prioritization and selection for a software release is a difficult undertaking. People have trouble comparing requirements of different types, and have natural biases towards certain types of requirements. This results in requirements selection decisions being made that are not in the best short and long-term interest of the product.

This paper proposes to address these problems by using portfolio theory. Portfolio theory is used to minimize the risk of an investment strategy for an expected return.

Portfolio theory can be applied to requirements prioritization and selection in a number of ways, but this paper proposes breaking requirements up by type, and using portfolio theory to determine how much time should be spent

on each type of requirements for a given release of the software. Portfolio theory or other methods can then be used to prioritize between the requirements of each type.

As many organizations do not have a deep understanding of the value and risk posed by individual requirements, this approach provides an easy way to systematically determine the which requirements will lead to a preferred and sustainable outcome.

The next step of this work involves supporting practitioners to understand, apply and evaluate the proposed approach. In the longer-term an evaluation should be made of the affect of using this approach on a software products bottom line.

REFERENCES

- [1] [1] C. Ebert, "Requirements BEFORE the Requirements: Understanding the Upstream Impacts," in 13th IEEE International Conference on Requirements Engineering (RE'05), Paris, France, 2005, pp. 117-124.
- [2] [2] B. Regnell and S. Brinkkemper, "Market-Driven Requirements Engineering for Software Products," in Engineering and Managing Software Requirements, 1st ed, C. Wohlin and A. Aurum, Eds. Berlin Heidelberg: Springer, 2005, pp. 287-308.
- [3] [3] A. Aurum and C. Wohlin, "Aligning Requirements with Business Objectives: A Framework for Requirements Engineering Decisions," in the Requirements Engineering Decision Support Workshop held in Conjunction with the 13th IEEE International Conference on Requirements Engineering, Paris, 2005.
- [4] [4] T. Gorschek and A. Davis, "Requirements Engineering: In Search of the Dependent Variables," Information and Software Technology, vol. 50, 2008, pp. 67-75.
- [5] [5] J. Azar, R. K. Smith, and D. Cordes, "Value-Oriented Requirements Prioritization in a Small Development Organization," IEEE Software, 2007.
- [6] [6] M. Lindgren, A. Wall, R. Land, and C. Norström, "A Method for Balancing Short- and Long-term Investments: Quality vs. Features," in 34-th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2008.
- [7] [7] T. Gorschek, S. Fricker, and K. Palm, "Star Search – Developing a Scalable Innovation Process," (in print) IEEE Software 2009.
- [8] [8] N. D. Fogelström, S. Barney, A. Aurum, and A. Hederstierna, "When Product Managers Gamble with Requirements: Attitudes to Value and Risk," In *Proceedings of the International Workshop Conference on Requirements Engineering: Foundation for Software Quality (RefsQ)*, Amsterdam, Netherlands, June 2009.
- [9] [9] N. D. Fogelström, Mikael Svahnberg, Tony Goschek; Investigating impact of business risk on requirements selection decisions; In the proceedings of 35-th EUROMICRO conference on Software Engineering and Advanced Applications (SEAA) Patras, Greece, August 27-29, 2009.
- [10] [10] G. M. Constantinides, A. G. Mallaris, "Portfolio theory" in Handbooks in operations research and management science, vol. 9, R. A. Jarrow, V. Maksimovic and W. T. Ziemba, Eds., Elsevier Science, New York, 1995.
- [11] [11] N. D. Fogelström, Understanding and supporting requirements engineering decisions in market-driven software product development, Blekinge Institute of Technology Licentiate Dissertation Series, 2010.
- [12] [12] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing requirements," IEEE Software, vol. 14, pp. 67-74, 1997.
- [13] [13] S. Sivzattian, and B. Nuseibeh, "Linking the selection of requirements to market value: A portfolio-based approach," In Proceedings of 7th International Workshop on Requirements Engineering: Foundation for Software Quality (RefsQ), Interlaken, Switzerland, 4-5 June 2001.
- [14] [14] S. R. S. Sastri, S. Mohanty, K. K. Rao, E. J. Elton, and M. J. Gruber, "Modern portfolio theory, 1950 to date", Journal of Banking and Finance, vol. 21, no. 11, pp. 1743-1759, 1997.
- [15] [15] E. J. Elton, M. J. Gruber, S. J. Brown, and W. N. Goetzmann, Modern portfolio theory and investment analysis, John Wiley and Sons, Hoboken, 2003.
- [16] [16] E. Numminen, "Why Software Platforms Make Sense in Risk Reduction in Software Development – A Portfolio Theory Approach," in Proceedings of the 4th European Conference on Information Management and Evaluation, 2010.
- [17] [17] T. Copeland, and V. Antikarov, Real options: A practitioner's guide, Texere, New York, 2003 .
- [18] [18] M. Amran, and N. Kulatilaka, Real options: Managing strategic investments in an uncertain world, Oxford University Press, New York, 1989.
- [19] [19] A. K. Dixit, and R. S. Pindyck, Investment under uncertainty, Princeton University Press, Princeton, New Jersey, 1994.
- [20] [20] C. Shapiro, and H. Varian, Information rules – A strategic guide to the network economy, Harvard Business School Press, Boston, 1999.
- [21] [21] E. D. Beinhocker, "Robust adaptive strategies", Sloan Management Review, vol. 40, no. 3, pp 95-106, 1999.
- [22] [22] C. Veorhef, "Quantitative IT Portfolio Management," Science of Software Programming, Vol. 45, No. 1, pp, 1-96, 2002.
- [23] [23] K. Schwaber, Agile project management with scrum, Microsoft Professional, 2000