HELSINKI UNIVERSITY OF TECHNOLOGY

Mat-2.177 Seminar on case studies in operation research

Nokia Networks: Product release definition

Final Report

Samu Kyrklund	76907S	(Project Manager)
Johanna Ahtola	54347S	
Eino Laitinen	57726E	
Tomi Lähdemäki	57815V	
Maria Routti	60021A	

April 21, 2006

Abstract

Authors:	Johanna Ahtola, Samu Kyrklund, Eino Laitinen, Tomi Lähdemäki, and Maria Routti	
Title:	Product release definition	
Date:	2006/04/21	**Number of pages: 2 + 50**

This study was done as a part a course operations research project seminar in Helsinki University of Technology and the target company was Nokia Networks which is one of the leading mobile network manufacturing companies.

The objective of the study was to build a tool that can be used to define features that should be included in a specific product release. However, also R&D processes are discussed shortly in general level and product release process models are investigated and compared to the current situation in Nokia Networks.

A product release consists of a selection of possible features that presents different functionalities in the final product. These features have a value, a cost in hours and they may have different kinds of interdependencies. The values may not be accurate and intervals for these can be used. A release project has also a strict hour budget.

As a result a tool for Microsoft Excel was created. The tool uses robust portfolio modeling to select features that should be included in a product release. Also other software platforms were investigated and evaluated for the purpose of the tool. The developed tool is a rough test version and we suggest that a new project to develop a better user interface should be implemented.

This report can be also used as a manual for the tool.

Keywords:	Product release, Features, R&D, Robust Portfolio Modeling, Nokia Networks

Table of Contents

1 Introduction

1.1 Background of the research

The client for whom this research project is being conducted is Nokia Networks Ltd. Nokia Networks is a leading provider of network infrastructure, service delivery platform and related services to mobile operators. This project is being done for the R&D unit. The aim of this project is to develop an optimization tool for selecting the contents of a product release.

1.2 Motivation of the research

This research project is part of the course Operations research project seminar arranged by the Systems analysis laboratory at the Helsinki University of Technology. Our assignment was to develop a tool for selection of release content for Nokia Networks. These releases are generally very large with strictly defined budgets. This specific release contains two separate products, which are combination of software and hardware. The content is defined by features, which in this context mean new product functionalities. The features may have dependencies and some features may have value only in conjunction with another feature. Usually a feature has an impact on both products. This makes the selection of features a complicated problem that needs an optimization tool in order to find the optimal solution. This tool is intended for supporting the decision making in addition of other techniques and approaches.

1.3 Objectives of the research

The goal set for us on behalf of Nokia was to develop a decision support tool to help in the selection of features for each release. The tool was developed but the scope of research was widened to include also the improvement of the release process. The process was included to avoid the so called “garbage-in garbage-out” phenomenon. The meaning of this phenomenon is that no matter how good a decision support tool is the output can’t be of any value if the input data is not valid and based on real figures and

reliable processes. The inclusion of the process view was considered as an important aspect to ensure the output quality of the tool. The release process is developed based on the current process and the alternative approaches found through a literature review.

1.4 Research scope

The objective of this research was to develop a test version of a tool that selects the most suitable features to each release. Because test data was not available, the quality of the tool could not be verified. It has only been tested using generated data of the same structure as would potentially be available in the future at the client. The tool is therefore mainly a test tool that provides one way of selecting the features. Because of the limitations of this tool, other tools were also explored in the literature review. The aim is only to introduce these methods, not to apply them to Nokia's case. Also the usability of the tool is left to minor consideration and the user interface is based on filling matrices in MS Excel.

The improvement of the release definition process was also included. The discussion is based on current literature and offers alternative approaches to R&D and release processes. The discussion related to the R&D process concentrates on the future trends of R&D management. It does not give suggestions about how Nokia should change their R&D process but provides a possibility to benchmark the current situation comparing to R&D generations. Further the performance measuring of R&D unit is not discussed. The release process is examined first with the framework presented in the literature review and further through exploring available tools for requirements engineering and release planning.

Key concepts:

Stakeholder: Anyone who is materially affected by the outcome of the project (Ruhe 2003), e.g. customers, employees, subcontractors etc.

Requirement: A condition or capability needed by a user to solve a problem or achieve an objective (Kauppinen, 2005 s.12→ IEEE standards)

Feature: Service to fulfill one ore more stakeholder needs, normally feature implements multiple business requirements

Release: New version of software with new features implemented is brought to the market

Release process: The process of defining the content of each release, through requirements development to release content definition

2 Literature review

The role of R&D is generally acknowledged in management levels as strategically important. The efficiency of R&D is seen as a requirement for maintaining long term competitiveness. This efficiency in turn requires that resources are used efficiently implying that the focus should be on essential and value-adding activities. Competitiveness can be best supported by directing the development according to company goals by leveling the business strategy to R&D. This also directs towards what is considered as essential when allocating the scarce resources. (Cooper et al., 2001)

However the allocation between different activities is mostly done at the product management level. At this level methods and techniques are needed to assist in determining where to invest the resources to provide maximum value. The following literature analysis examines first the future of R&D and recognizes important aspects that need to be considered at the management level. Second part focuses on the release process at the point when product management has to choose the essential activities among all possibilities.

2.1 Research and development process

2.1.1 Evolution of R&D to current situation

R&D management has developed as the business environment changes. Decades ago the role of R&D was to develop technologies and inventions separately from business activities. It was mainly an overhead cost lacking a strategic framework. Since then it has evolved towards more collaborative approach. (Blomqvist et al., 2004)

When the business needs were taken as the bases of the development process the second generation evolved. However the management was project-by-project based and the prioritization of projects within and across businesses was found hard since there was no common goal for R&D.

In the third generation the collaboration among managers from different functions enabled meaningful direction of development instead of the shattered portfolio of R&D projects. The establishment of technology portfolios, roadmaps and lifecycle considerations linked the R&D to overall business strategy. Most companies run their R&D according to these principles nowadays. (Blomqvist et al., 2004)

2.1.2 Future of R&D

The trend is towards networked innovation management where the whole supply chain is involved in the development process. This approach presumes the industry structure to be more dynamic. "The scope of innovation management is broadened to include not just products and processes but business and market models that encompass the management of knowledge, technology and market/industry infrastructure."

The R&D process can be seen in the future as a networked innovation process that comprises greater overall organizational and systems integration and broader horizontal networking. (Blomqvist, 2004)

2.1.3 Benefits of networked R&D

Companies' competitiveness is increasingly originating from the non-technological issues. The competitive edge reached with technological core competencies may diminish and the sources of competitive advantage need to be understood more broadly. However the technological competence is the key as long as the focus is on providing value for the customer. R&D should continuously develop superior technologies and products to offer customers better value than the competition does. Continuous innovation requires vast amounts of new knowledge and this can be obtained through networking. Networking also shares risks and costs and collaborating with appropriate party can ease the integrating of short and long term time horizons.

2.1.4 Towards customer needs: R&D and marketing interface

The customer oriented approach in R&D should be adopted in order to sustain competitiveness. In some cases, especially in customer specific products, the customer is addressed straight and the customer's needs can be found out by asking before the product development. In new product development and in less customized products the needs have to be found out elsewhere and marketing department is a good source of information concerning customers' and market's needs. The collaboration should be two-way between marketing and R&D. Marketing should analyze customer and market needs and communicate them further to R&D and the R&D should keep the marketing aware of the new products under development. This way it can be designed beforehand when the new product should enter the market and with what magnitude. The researches show that successful communication between marketing and R&D is a critical success factor in new product development.

The customer needs should be the main source of system requirements and therefore the R&D should be closely interacting with the party that operates in the customer interface. Usually there are sales and marketing people who therefore possess also the greatest knowledge of market and customer needs. (Jin, 2001; Wang et al., 2002) However the common goals are hard to find. People in marketing have different goals and priorities when defining the essential activities. Overall the marketing and management consider the role of R&D product management as less important than other branches of management (Cooper et al., 2001). Marketing people are mostly interested in products that are likely to be a market success. R&D people are more interested in radical breakthroughs and "exiting" products. Another different view stems from time association: marketing may be more short term orientated than R&D. (Jin, 2001)

2.2 The Release Process

Decision processes are the driving force to organize a corporation's success. The decision process in question is the process of incremental software release. In the incremental software release processes, requirements are gathered in the initial stages and, taking dependencies and customer priorities into account, as well as the effort or cost required for each requirement, the system is divided into increments. These increments are then successively delivered to customers. There is a need to decide which requirements should be delivered in which release. (Ruhe, Greer, 2003)

The overall process manages requirements continuously while developing releases (see Figure 1). Activities in the process are: elicitation, requirements selection, construction, change management and documentation. The elicitation phase deals with the collection and initial classification of requirements. Requirements are described, named, and explained. The initial priority of the requirement is set. This phase is also called requirements engineering. The selection phase includes detailed specification of each requirement and release planning. In release planning the requirements to be implemented in the current release are selected and specified in detail. Validation is also an important part of selection. After selection the decisions are implemented and construction of the software is accomplished. Change management is active in parallel with construction and manages changes to requirements priorities. In the conclusion phase the process is documented and learning from history is initiated. (Höst et al, 2001)

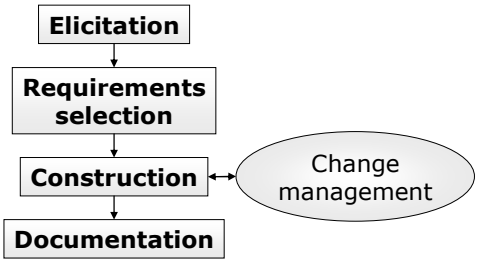

Figure 1. Process of software release (Höst et al, 2001)

A more comprehensive view of the overall process is presented by Amandeep et al (Amendeep et al, 2004). The process begins with characterizing and understanding the project environment. This includes determining the market demand for the product and

availability of resources in the organization. Next is the problem definition phase (see Table 1). This begins with identifying stakeholders and setting the relative importance of the different stakeholders. Elicitation of the requirements is part of this phase. The stakeholders input their views on the priority of the requirements also during this phase. The different constraints of the process are also identified. This can include resource constraints or interdependence of the requirements. After this, planning and execution of the problem solving follows. As a result the requirements to be included in the release are specified. An overview of the process is shown in Figure 2. Different methodologies for this solution exist and they will be reviewed later.

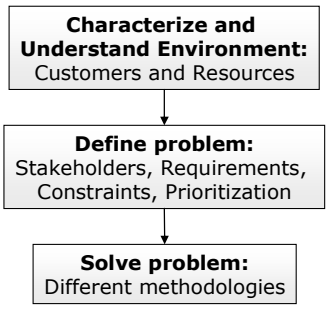

Figure 2. Process of software release (Amendeep et al, 2004)

Table 1. Activities in the problem definition phase (Amendeep et al, 2004)

Activities	
Identify the stakeholders:	
	sales representatives, users, investors, shareholders, project managers, product managers, developers
Elicit and specify the requirements:	
	information from different stakeholders
	business impact of requirements
Identify the relative importance of stakeholders	
Define the constraints	
	precedence, coupling or resource constraints (includes dependencies between requirements)
Stakeholders assign value to requirements	
	priority or financial value etc.

There exist more detailed methodologies to both requirements engineering and release planning. These will be reviewed in the next sections.

2.2.1 Requirements engineering

According to Kauppinen, the necessary requirements engineering activities are:

- Elicitation and analysis: discover user needs actively, identify most critical user needs
- Representation and analysis: set unique identifier for each requirement, apply use case method, prioritize requirements
- Validation: use different stakeholders to review requirements

(Kauppinen, 2005)

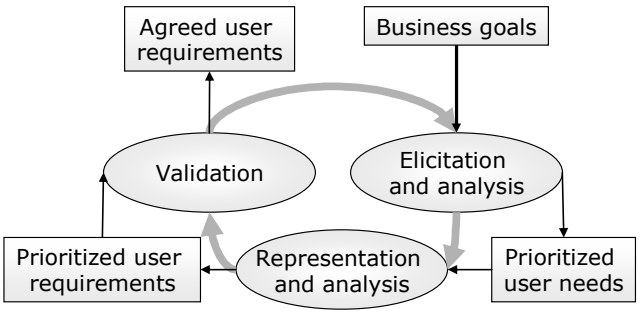

Figure 3. Requirements engineering process according to Kauppinen, 2005

A similar process is offered by Salo and Käkölä. Requirements are first captured, then categorized and refined, and last assessed by stakeholders. A follow-up activity is also included to allow for systematic monitoring of the requirements in release planning. These process descriptions however don't include methodologies for judging the relative priorities of the requirements. Different methodologies will be explained next. Cross-functional participation in the requirement engineering process is very important. (Salo, Käkölä, 2005)

Relative priority can be assessed through different attributes e.g. value, urgency, risk. Any one of these attributes can be used in the priority rating, depending on the business case of the product. The costs of the different requirements must also be discovered. Different techniques that can be used include Analytic Hierarchy Process, Quality Function Deployment, category grouping, etc.

A common and easy approach to prioritizing requirements is to group requirements into three priority categories. This is often done based on subjective estimates. Extending on mere groupings Wiegers proposes a semi-quantitative approach, which distributes a set of estimated priorities across a continuum. Customer value consists of both customer benefit

from present features and penalty paid if a wanted feature is not present. Cost and technical risk of implementing considered. This prioritization scheme should only be applied to negotiable features, not those that are top priority in any case. Typical participants in the decision are the project manager, key customer representatives, and development representatives. The steps are as follows:

- list all requirements at the same level of abstraction
- estimate the relative benefit of each feature to customer or business
- estimate relative penalty if feature not included
- total value is sum of benefit and penalty
- relative cost of implementing each feature
- estimate technical or other risk associated with each feature
- calculate priority number: priority = value / (cost + risk)
- sort the list according to descending order by calculated priority

This method is limited by the ability of the decision making group to estimate value, cost and risk and come to an agreement on these. (Wiegers, 1999)

Another method is the Analytic Hierarchy Process (AHP). In AHP the requirements are constructed into a hierarchical structure and a series of pair-wise comparisons are carried out to achieve a priority order. When doing pair-wise comparisons, the decision maker specifies his/her preference of one requirement to another and the global priority order is established by combining all the comparisons. This methodology has been extended by Lee et al (1996) by incorporating uncertainty of the preference comparison process and the dependence of requirements. Decision makers usually find that it is easier to give interval judgments than fixed value judgments on priority. By using the concept of comparison interval, a methodology based on stochastic optimization was introduced. The paper also captures another practical feature of the decision making process, which is dependence. In reality, there is often interdependence among the requirements. A technique for incorporating this through use of super-matrices, weight vectors and impact paths was introduced. (Lee et al, 1996)

It has, however, been noted that subjectively estimating cost and value using pair-wise comparisons of all requirements is impractical for anything more than a couple dozen requirements as the effort becomes prohibitive with larger amounts (Wiegers, 1999; Greer, Ruhe, 2004). More information on different prioritization methods can be found in Karlsson et al. (1998).

Software quality function deployment (SQFD) is a method of quantifying customer preferences. The basis is to obtain and translate the needs of the customer, in their own words, into a set of detailed design specifications that can be used to guide all phases of the production process. The application of QFD to the software development process is a means of formalizing the collection and transformation of customer needs into a set of system design specifications. Additionally, SQFD enables a development team to evaluate their design based on: quantified statements of customer priorities, explicit representations of the relationship between requirements and competitive assessment. The SQFD technique is uses a series of relationship matrices to elicit customer priority of requirements, relationships between requirements and features of competing products. The SQFD is robust and comprehensive and relates real customer value to product features. (Barnett, Raja, 1995)

Another approach to documenting the customer's needs is the Use Case –method. Use Cases consist of visual description of the relationships between stakeholders and processes, and a text document that details the transactions between each stakeholder and process part. Together with customer the customer's needs are specified through iteration rounds from business requirements to form detailed description about the input and output of the system. The function of the system is not described and it is considered as a black box from the customer point of view. The prioritization of the use cases is done at some point when the needs are sufficiently translated to more specified level. The prioritization methods described earlier can be used to select the suitable cases. After the chosen use cases are constructed in collaboration with the customer the system developers design the best way to perform the depicted processes and transactions. (Kulak et al., 2000)

There is a need for cross-functional and cross-company coordination in requirements engineering. To help in this, different groupware tools are available. Requirements engineering is the phase of the R&D process that is most amenable to groupware support tools and allows for the largest and cheapest opportunities to shorten the development cycle. The tool can help in establishing good communication and collaboration between different functional groups (e.g. development, marketing, sales, customer support). The groupware techniques present a sound methodology for requirement elicitation and stakeholder negotiation. (Salo, Käkölä, 2005; Amandeep et al, 2004)

2.2.2 Release planning

Release planning is the other important phase in the release process. This phase means selecting an optimal set of requirements for realization in a certain release. This selection task is normally preceded by requirements prioritization and resource estimation. At the time of prioritization it is difficult to be fully aware of context and circumstances. Hence further judgment is needed. In this phase requirements engineering for market-driven software development meets the market perspective. This phase is seen as a major determinant of the success of the software product. (Carlshamre, 2002)

2.2.2.1 Intuitive release planning

The problem that needs to be solved in release planning has been seen as an ill-defined, wicked problem (Ruhe, Saliu, 2005; Carlshamre, 2002). This means that: there is no stopping rule for optimization; there are better or worse solutions, but no optimal one; every problem essentially unique; no objective measure of success exists. Human intuition and capabilities can be used to clarify the problem. Approaches exist that are based to addresses implicit and tacit aspects of release planning. These approaches lack an emphasis on formal process and optimization. Rather they are based on expert opinion, and use of tacit knowledge.

One of approach is agile development, which leverages the advantages of small, iterative software releases to receive early customer feedback, and also takes this more humanistic approach. Release planning focuses on planning for the next iteration. The planning

procedure relies on meetings between the important stakeholders to discuss and negotiate informally which features to develop next and how much effort they require. Exponential growth in the number of possible release plans surpasses the power of manual plan generation, especially as the number of features and stakeholders grows. (Ruhe, Saliu, 2005)

A study of the reality of release planning concludes that the planning is focused on guessing or sensing, rather than creative choices and then systematically evaluating them. Typically the process of release planning is a manual effort supported by spreadsheet computations. Tough decisions have been made manually on the basis of experience, tacit knowledge or feelings of the development personnel. Some requirement prioritization practices include: assessing the value of requirements for customers and the development cost early in the development phase without formal methods; building priority list of focal areas; using prioritization scales by dividing requirements in categories; negotiation in project meetings, which involves mutual discussions and agreements. When using these manual approaches, it may be difficult to consider all the constraints and not enough time and resources to generate different scenarios. (Momoth, 2004)

2.2.2.2 Mathematical optimization approaches

Some researchers have modeled the problem as a specialized optimization problem. In formulating an optimization model for release planning, Bagnall et al assign weights to customers according to their importance to the software company. The objective is to find a subset of customers whose features must be satisfied (within the budget) (Bagnall et al, 2001). Similarly, Jung's approach selects features that give maximum value for minimum cost, considering the software system's budgeted costs (Jung, 1998). These optimization approaches cope better with larger problem sizes, but they don't give customers an opportunity to participate in release planning decisions, and they don't plan beyond a single release. (Ruhe, Saliu, 2005; Saliu, Ruhe, 2005)

2.2.2.3 Hybrid approaches

Empirical findings have underlined several challenges in release planning. Requirements have interdependencies that need to be incorporated into problem solving. Stakeholder participation is increasingly important to ensure market demand. Planning approaches need to be flexible and allow for when changes in requirements and their priorities occur. Resource constraints, such as skills and time of development employees need to be taken into account in scheduling a release. Corporate strategy and business goals should be involved in the decision. A set of criteria for a decision support system attained by Momoth through empirical research is:

- supports decision makers rather than replaces them;
- utilizes data and models;
- solves problems with varying degrees of structure and non-structured tasks;
- focuses on effectiveness rather than efficiency of the decision process.

(Momoth, 2004)

Findings by Carlshamre on the design implications for new tool for release planning decision support are similar. Important points are:

- Should be interactive, and support exploration and direct manipulation. An algorithm for release planning should be available.
- Information should be presented according to different models, e.g. timeline, graph, tree structure
- Comparison of requirements, release suggestions and arbitrary groups of requirements should be supported.
- Must support planning several consecutive releases.
- Algorithm needs to consider existence and strength of interdependencies.
- The algorithm does not need to take into account more than a few parameters: value, cost, dependencies. Although there are many parameters that affect the decision, not all are known in advance, or they cannot be elicited.

- The support tool should act humbly: should not pretend to be exact. The planner must have full control over the interaction.

It is concluded that comprehensibility is more important than exactness, and presentation more important than calculation. (Carlshamre, 2002)

Based on these empirical findings it is obvious that the decision process requires a support tool that allows for human intelligence, but also has computational power to deal with a large number of interdependent requirements. Two different tools exist in literature that allow for this. These solutions are a hybrid between informal methods and mathematical optimization tools.

A group of researchers including Ruhe, Greer and Saliu have developed a decision support tool (EVOLVE) for release planning, which provides 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. Stakeholders are involved in the construction of requirement values and urgency. The variables optimized are thus release total value and urgency of requirements included. Advantages of this method over existing methods include:

- takes into account stakeholder priorities as well as effort constraints for all releases.
- recognizes that software requirements are delivered in increments.
- considers inherent precedence and coupling constraints. Existing approaches do not cater for dependencies between requirements.
- uses a genetic algorithm means that the final release plan arises from a population of solutions. This allows those solutions that break constraints to be disallowed without deterioration of the method.
- offers greater flexibility by allowing changes in requirements, constraints and priorities.
- recognizes that stakeholders have priorities for requirements that may be conflicting.

- recognizes that there is a negative impact penalty of delivering requirements in a sequence contrary to a stakeholder's priority and a positive benefit of delivering high priority requirement earlier.
- not all stakeholders are treated equally, so that the effect of their input can be weighted.

The approach formulates a series of problems as variants of the original formal model. Then these problem variants are solved to generate a set of qualified alternative solutions. A human decision maker—such as the project manager—evaluates the solutions based on his or her experience and familiarity with the problem context. In this way, art and science complement each other. The different phases of the decision process are exhibited in Figure 1.

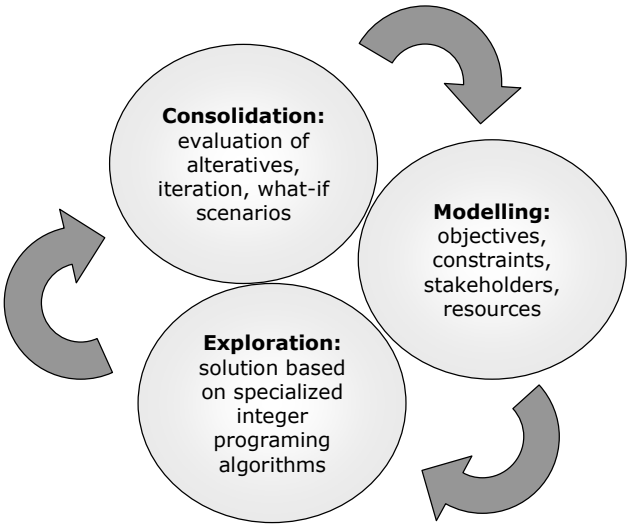

Figure 4. The iterative process of release planning

In modeling the problem is conceptualized. The objectives and constraints are planned, stakeholder voting on priorities is arranged, and estimates of the likely amounts of various resources needed are determined. In phase 2 the solution plan is generated based on the formal model. Specialized integer programming algorithms are used to explore the solution space and generate solution alternatives. In the consolidation phase the decision maker evaluates the c solution alternatives based on experience and the problem context. Then, if need be, he or she can modify parts of the underlying model or make some local

decisions (perhaps pre-assigning some features to specific releases). (Ruhe, Saliu, 2005; Amandeep et al, 2004; Ruhe, Saliu, 2002; Greer, Ruhe, 2004; Ruhe, Greer, 2003)

Another comprehensive approach to decision support in the release planning process is offered by Carlshamre (2002). His Release Planner loads current requirements, and sorts them in order of value. The decision maker enters resources available for design, implementation and basic test. The program then presents five releases ordered by total value. There is a possibility for the planner to delay some requirements or force the inclusion of some. The is modeled as a binary knapsack problem, and solved using a modified greedy algorithm. This approach does not guarantee an optimal solution, but finds a number of good ones very quickly. Interdependencies of different types included: "and", "requires", "affects customer value of", "affects implementation cost of". In the current version however, only "and" and "requires" are always obeyed, others require decision maker judgment. (Carlshamre, 2002)

3 Current release definition process

The current process of release definition is presented in Figure 5. The process starts from the release specific budget that sets the most important resource restriction to release scope when selecting the features. The second restriction is the other resources formed by those of R&D personnel, who are involved in the actual release development process. When the availability of needed skills and working hours are determined the next step is to determine the requirements and map them to features. One feature consists of several requirements which are all fulfilled by the one feature. After the potential features are compiled from the requirements they are valued with measures like short and long term price. When the restrictions and the possible features to be included in the release are valued, the iteration of the suitable features can start. In this part the tool constructed as the output of this research gives an optimal solution based of the input data. This solution may be used as guidance when selecting the features. The data needed in the selection has to be valid and available. In addition of the optimization tool ad hoc methods are recommended to use.

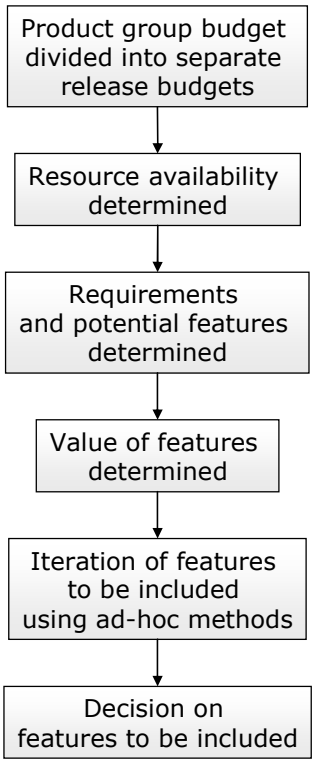

Figure 5. Current release definition process

3.1 Valuation of features

The valuation is done by the product management based on a price range given by the sales and marketing department. The range defines minimum and maximum values and the most likely value lies somewhere between them. The estimation of the most likely value is currently based on the expertise and intuition of the product management persons. Testing of the feature with some customer would produce more accurate estimates for the values and this approach might be considered in the future.

3.1.1 Short term and long term values

The valuation is divided to short term and long term values. The short term value is based on estimates like how much would the customer pay for certain feature or how many customers would benefit from it whereas the long term value measures are more strategic. Long term value can be estimated through estimates on market share increase or decrease of overall production costs. Typically the features that have more short term value are

customer orientated features, or fixes or improvements to current functionalities whereas the features with more long term value are typically related with software architecture improvements or other features that take more than one release to create value. Because the measures of long term value are based on wider strategic goals the valuation is very challenging. In addition the follow-up of long term values is probably very challenging. Short term values can be followed with more reliable measures like license keys which tell the number of customers using the feature in question. The process is still in ramp-up phase so no follow-up is performed currently.

3.1.2 Budgeting

The budget is release specific and is translated to workload estimate. These working hours should be then divided to such features that optimize the created value within the maximum workload set by the budget. The division of budget between features creating long term and short term value is guided from higher management levels who determine the strategic guidelines. The profitability targets are set product wide and are not aggregated to release or feature levels. In practice the budget boundaries are not very strict because of the estimations of workloads. However the tool assumes that it is binding. This was based on the wishes of the client that recommended this approach. It was said that minor budget overruns can be allowed but major exceeding is always an exception and renegotiations has to be made regarding the budget.

3.1.3 Pitfalls in selecting the features

The main problem concerning the tool is the availability of the data. The location is distributed and the collection might turn out harder or slower than expected. Also the estimating methods of the values are mainly based on educated guesses and intuition. Selecting features based on intuition is a common pitfall because so called “pet projects” can be chosen without adequate justification. Another common mistake is the lacking of strategic criteria. In this process the long term value represent the strategic part and is thus taken into account. Without this the features with short term value could be emphasized based on low resource consumption and near future cash flows. It is critical to keep the balance between short term and long term features in order to sustain

competitiveness. The problem of selecting low value features is basically removed by the optimization tool that shows those features which should not be included at any case. Also the estimated workloads for each feature prevent the resources from being overloaded. Only if the needed effort turn out to be underestimated the resources can be stressed with too many features at hand. If some feature is discarded from current release it can be selected to the next release. However the valuation has to be performed again to prevent inadequacy of the input data. (Cooper et al., 2001)

4 The problem definition

4.1 Description of the problem

The problem is how to select an optimal combination of features to be developed for the next product release. The optimal condition is to maximize the sum of feature's short and long term values in a release. All features have both upper and lower values.

The problem is constrained by the budget of total hours. Development of each feature requires certain amount of work hours. There is selection of about a hundred of different features from which around 20 will be selected for certain release. Furthermore, between the features there are interdependencies. Basically four types of dependencies can be recognized:

1. Mandatory. Selection of certain feature requires another feature(s) to be selected. This interdependency may be asymmetric meaning that advanced feature may require some basic functionality to work. However the basic functionality doesn't require the advanced one. Mandatory constraint may also be independent of others that certain feature has to be selected to the release.
2. Mutually exclusive. Two features cannot be selected for the same release.
3. Amplifying. Choosing two features amplify each other creating greater value together than just summing up them individually.
4. Attenuating. Choosing two features attenuate each other creating lesser value together than just summing up them individually.

4.2 Mathematical formulation of the problem

The problem can be formulated as a linear programming problem. This is important because there are many effective tools and algorithms for linear problems. This problem is a special case of linear problem because all the decision variables are binary variables so they can get only values 1 (selected) or 0 (not selected).

A normal form binary linear problem is

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} v_i z_i \\ s.t. \quad & Az \le b \\ & z \in \{0,1\} \end{aligned}$$

Here the z vector includes n decision variables and v_i:s are corresponding values of the particular decision made. The constraints of the maximization problem are presented in the form $Az \le b$ where A is a m x n matrix including all the costs of the decision variables in m constraints. Vector b includes the corresponding limits and it has m values.

This kind of problems is known as knapsack problems and they are widely investigated in the LP-literature.

4.2.1 Variables

Each decision variable describes the selection of one specific feature into the product release or portfolio. If the decision variable z_i gets value 1, the feature no i is selected into the product release and value 0 of the decision variable presents that this particular feature is not included into the portfolio. There are as many decision variables as the total number of different features that can be selected into the product release. Portfolio p can be depicted with single vector $p = [z_1,...,z_n]^T$.

4.2.2 Objective function

In this problem definition there are several decision criteria. We have to take into account both short- and long-term values of the features in the decision making. This makes the objective function a little bit more complex. First we have to define the relative weights of the criteria. Relative weights can be defined in the following way; if value of first criteria grows one unit what is the equal change in the value of the second criteria? Let us say that the equal change is a unit in the second criteria. Then the relative weight of the first criteria is $w_1 = \frac{1}{1+a}$ and the relative weight of the second criteria is $w_2 = \frac{a}{1+a}$.

These weights have a characteristic that they sum up to one and are always greater than zero:

$$\sum_{i=1}^{2} w_i = 1$$
$$w_i \geq 0 \quad , \forall\, i$$

The objective function can now be written as:

$$\sum_{i=1}^{n} z_i (w_1 v_{i1} + w_2 v_{i2}) = V(p, w, v)$$

This differs from the original one only by having the weighted sum of the values of the decision variables in the place of single value in the original objective function.

4.2.3 Constraints

Constraints are equations that limit the feasible set of possible solutions. In this problem there are four types of different constraints.

1. *Budget constraint*:

   All the features have a rate of hours that they need to be developed into the product. Furthermore, the whole product release project has a budget of total hours that can be spent to the product release. We define the hours required for a i:th feature as c_i and the budget limit of the hours is b. Thus the constraint can be written as:

   $$\sum_{i=1}^{n} c_i z_i \leq b$$

2. *Mandatory constraints*:

   Some of the features are not possible to include into the product release without one or more other features. These constraints can be formulated as linear constraints in a following way:

   $$z_i - z_j \leq 0$$

   Here feature i can not be selected without feature j but j can be selected without i.

3. *Mutually exclusive constraints*:

   There are also features that can not be included into same product release, so they are mutually exclusive. Theses constraints can be formulated as:

   $$z_i + z_j \leq 1$$

   Where feature *i* and feature *j* can not be selected into the same product release,

4. *Amplifying and attenuating constraints*:

   The last constraints are most complex. These are constraints that describe a phenomenon where choosing two or more specific features into the product release they create more value than just the sum of their individual values. To model this characteristic we have to add a *dummy* feature into the portfolio of all possible features. This dummy feature has values (long- and short-term) that are equal to the extra values generated when a specific selection of features are included into the product release. The cost of this dummy feature is, however, zero. Furthermore, we need to constraints. The first one ensures that if all the features required to the extra value are selected (let's say *i*:th and *j*:th features), also the dummy feature will be selected. This can be written as:

   $$z_i + z_j - z_{dummy_k} \leq 1$$

   This constraint is for *k*:th dummy feature.

   The second constraint is needed to secure that the dummy feature can not be selected when *i*:th ja *j*:th features are not selected. The following formulation presents this:

   $$z_{\text{dummy}_k} \leq \frac{1}{2}\left(z_i + z_j\right)$$

All these constraints can be hereafter presented in a simple matrix form $Az \leq b$ that is standard LP constraint form.

4.2.4 Uncertainty of the values

The short- and long-term values vi1 and vi2 may include some kind of uncertainty so that the exact values are not possible to define but both minimum and maximum values can be approximated for all feature values. These minimum and maximum values are defined as v_{i1}^{+}, v_{i1}^{-}, v_{i2}^{+}, and v_{i2}^{-}.The probability distribution between these maximum and minimum values is not specified. Only thing that can be said is that the value is between min and max.

4.3 Solution methodology

4.3.1 Dominating portfolios

There is not always possible to select the portfolio that is the best in any realizing case, because the values of different features are not fixed. Each feature's value can be any in the given interval. The dominating portfolios are those that are in some case the best portfolios.

Mathematical definition of the domination is following. Define two portfolios p and p'. When p dominates p' the following is true

$$\begin{cases} V(p,w,v) \geq V(p',w,v) & \text{for all } v \in S \\ V(p,w,v) > V(p',w,v) & \text{for some } v \in S \end{cases},$$

where S is set of all feasible values for each feature in given interval.

Instead of finding the dominating portfolios it is easier to test whether one dominates the other, non-dominating portfolio. Non-dominating portfolios should never be selected.

Dominance between two portfolios can be easily checked. This is simply done by removing matching features from both portfolios and comparing total value of remaining features. If maximum value of remaining features in the first portfolio is smaller than minimum value of remaining features in the second portfolio, the first portfolio is dominated. (Liesiö et al. 2006)

4.3.2 Corevalues

After all the dominating portfolios are found there maybe result of hundred different possible combinations of features for product release. Corevalues are used to help decision maker to choose the right features. Each feature's corevalue ranges from 0 % to 100 %. The value expresses percentage of how many of dominating portfolios include this certain feature.

Corevalue is simply calculated by summing up all dominating portfolios $p_i = [z_i^1, ..., z_i^n]$ and dividing by total number of dominating portfolios. Corevalue of j-th feature is

$$c_j = \sum_{i=1}^{k} \frac{z_i^j}{k},$$

where the k is the number of dominating portfolios.

If features corevalue is 100%, it should be selected in any case. On the other hand if corevalue is 0, it should never be selected. Anything in between is left to the decision maker to decide.

4.4 Different solution methods

We have chosen robust portfolio modeling (RPM) as a solution method that is used to solve the mathematical problem described above. However, RPM methodology is not used in a pure sense because it is very complicated to adapt it into commonly used office software. This problem is not a complete RPM problem because the weights of the decision criteria are fixed in this case while RPM uses incomplete information to describe those. Here is described three possible ways to solve the problem using different variants of the RPM method and different software.

4.4.1 Pure RPM and independent software

Robust portfolio modeling is an extension to preference programming and it makes possible to solve portfolio problems when there is incomplete information on projects (in the context of this work projects are equal to features) and/or decision criteria weights.

The method uses a calculation of non-dominated portfolios to classify projects into categories related to their efficiency. The simplest way to calculate non-dominated portfolios is to calculate all possible portfolios and take all feasible portfolios from these and then by a pairwise check rank out all dominated portfolios. The problem is that the number of possible portfolios is 2^m. Where m is the number of projects that could be selected. The calculation of portfolios from 40 projects might take up to 12 days (Liesiö et al. 2005) with a normal computer.

Leisiö et al. (2005) suggest a dynamic programming algorithm to solve all non-dominated portfolios. This algorithm starts building portfolios from an empty portfolio. After adding one project into the portfolio the algorithm checks if this portfolio is already dominated by any earlier portfolio. Only portfolios that use resources efficiently are stored for a later use. After all non-dominated portfolios are calculated the project can be divided into three categories. These are projects that will be chosen into the optimal portfolio even if additional information about decision criteria values or weights is added, projects that will not be selected in any case and then other projects that can be chosen if additional information is gathered (Figure 6).

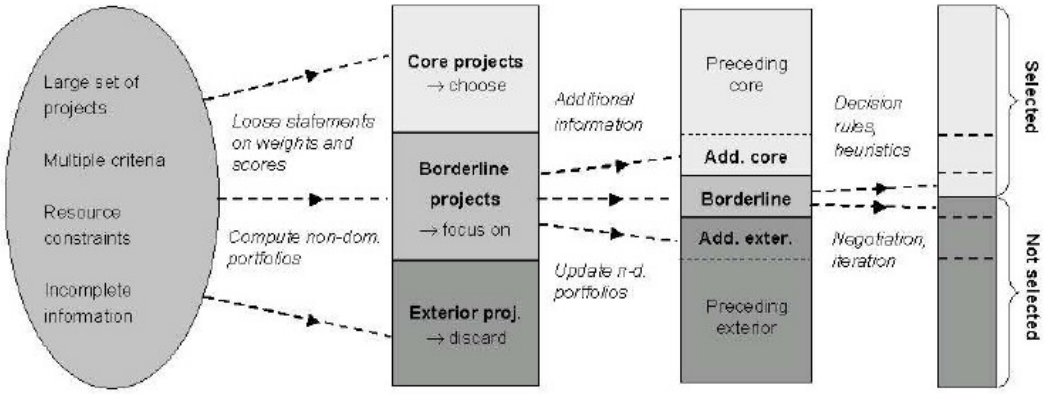

Figure 6. Classification of the projects in RPM context (Liesiö 2005)

There exists very few software that uses RPM on a full scale. One software was found from the Systems Analysis Laboratory in Helsinki University of Technology but this software was still in a development phase and can not be used outside the laboratory. We, however, got a possibility to try the software to solve our example product release.

4.4.2 RPM and Monte Carlo simulation

Because of the complexity of the pure RPM methodology a simplified method must be developed. We found out that in the Systems Analysis Laboratory in HUT there has been experiments using Monte Carlo simulation to solve non-dominated portfolios approximately.

This algorithm takes value randomly from a uniform distribution between the lower and upper bound of the project value. Then it simply calculates an optimal portfolio for these particular circumstances. The optimization problem is now an original knapsack problem that can be solved as a integer linear programming problem (ILP). This portfolio has to be non-dominated because it is the best in these circumstances. After optimal portfolio is calculated the algorithm checks if this is a new portfolio and saves it. If the portfolio is same that is found earlier it is not saved and algorithm starts from the beginning. This algorithm has to be repeated in many times to find out as many as possible non-dominated algorithms. There are however a couple of problems with this algorithm. First, the number of repetitions has to be very large because usually the algorithm finds same portfolio many times and secondly we can never be sure that all non-dominated portfolios are found.

This algorithm was also tested to find optimal tool for our problem context. This was implemented using MatLab software and a simple code that we got from the Systems Analysis Laboratory in HUT. Also an external linear programming solver was used to solve efficiently an optimal portfolio in every round of simulation. We also thought to implement this algorithm into Microsoft Excel and Visual Basic environment but it was abandoned because LP-solvers in these environments are not efficient enough to solve tens of thousands problems in reasonable time.

4.4.3 RPM and top portfolios

The third method or version of RPM that we developed to solve the problem is based on the following algorithm.

First, a portfolio that maximizes the minimum value of the portfolio is calculated. This is called MaxMin portfolio. After that a portfolio that maximizes the maximum value of the

portfolio is calculated and this is stored. Then we try to find portfolio that has the second best maximum value. This has to be at least little smaller than the best value. This stage is then repeated to find always the next best portfolio until the maximum value of the next best portfolio is below the original MaxMin value. All the portfolios can be calculated using ILP methodology to solve a knapsack problem. This calculated set of portfolios surely includes all non-dominated portfolios (the calculated MaxMin portfolio dominates all the rest portfolios) but it may also include dominated portfolios. After all these portfolios are calculated a pairwise check to all selected portfolios is performed to discard possible dominated portfolios. In this context a pairwise check is only needed to compare each portfolio to portfolios that have bigger maximum value.

We implemented this algorithm into MS Excel using Visual Basic language. However, the Excel's normal solver is not the most efficient one and it can be used only for small problems (approximately having only 10-20 projects). For bigger problems a more efficient LP-solver is needed. We used a trial version of Premium Solver for Excel[1] but there are also other commercial solver add-ons (and even more efficient) for Excel.

[1] Premium Solver for Excel's trial version (15-days) may be downloaded from www.solver.com

5 Product Release Optimizer - tool

5.1 Tool description

PRO 1.0 (Product Release Optimizer) tool was created to optimize product features for new product release. PRO 1.0 is able to evaluate from maximum of 100 given features the best portfolios within the budget limits. Every feature has a given cost and value. Value is divided in short-term value and long-term and for both values can be given an interval where maximum and minimum values are defined. Long and short term values can be weighted differently. Features can also have four types of dependencies that were described in problem definition chapter. There can be up to 30 mandatory conditions, 30 mutually exclusive conditions and 10 amplifying or attenuating conditions. When the budget and the features are defined, tool calculates core values for all of the features which are shown in a graph. In addition, all the non-dominated portfolios that were founded can be viewed.

5.2 Motivation for the tool development

At the beginning of the project there was discussion with the client Nokia Networks creation about creating new tool to help with the decision making in the product release definition process. The wish of the client was the tool would be created using MS Excel. At first Excel did not seem to be a suitable program to be used in solving this kind of problem including lots of features with different dependencies. However after the test data was generated, we decided to try if the Excel could handle a simplified problem including only 20 features with few dependencies. In the first version of the tool the problem was not formulated as a linear problem and the Excel solver got different results depending on the starting values of binary variables. Also the solving of even one portfolio took a lot of time. The crucial point in developing the tool in Excel was reached when the problem was formulated all over again as a linear problem. After this the results were no longer dependent on the starting values and calculating of the portfolios was much faster. After defining the problem again it seemed that after all the Excel could be used when creating the new tool as the client had wished.

5.3 Steps for solving a problem

Solving a new problem with PRO 1.0 tool can be divided to eight steps showed in Figure 7. First step is to clear all the old values so that the new features will not include old information that could give wrong results. All the old dependency values should be also removed. Next steps are to define budget, value weights and variation. Defining variation can be done either manually or using percentages. Fourth step is to define all the features that could be included in new product release. A certain cost and value is assigned to every feature. After defining all the features and their costs and values the next step is to define all dependencies across features. This is done in three separate sheets that include mandatory dependencies, mutually exclusive dependencies and dummy variables. Last steps are to use macros to calculate core values, make decisions based on the results and choose which features should be included in the new product release.

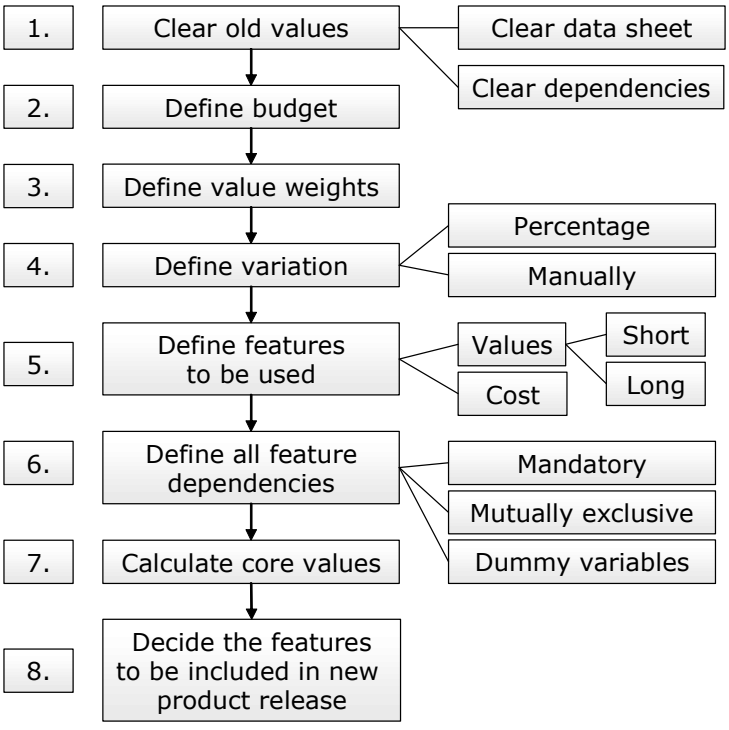

Figure 7. Steps for solving a problem

5.4 Filling the data sheet

In this chapter is explained how the date sheet should be filled out. Budget cell, different features with values and value weights are shown in Figure 8.

	A	B	C	D	E	F	G	H	I	J
1										
2				Budget:	160		Value weights		Short:	1
3									Long:	0
4										
5										
9		Feature name	Feature number	Cost	SHORT VALUE	Short value min	Short value max	VALUE LONG	Long value min	Long value max
10		name1	FN1	17,92		9,97	10,89			
11		name2	FN2	23,08		6,84	7,27			
12		name3	FN3	24,24		2,5	2,77			
13		name4	FN4	16,78		5,47	5,92			

Figure 8. Data sheet

5.4.1 Budget

Budget is the amount of money that can be used for the features included in the new product release. All the features that are included in a solution have a certain cost and the sum of these costs will never exceed the budget. Budget could also be determined as a work hours if the costs of features are given in a same way.

5.4.2 Weights

Value weights determine how the user wants to weight short-term values and long-term values of the features. One weight can be any number between 0 and 1 so that together both weights sum up to one. Only short-term weight needs to be changed and the long-term weight is calculated automatically. As in Figure 8 if short term weight is determined as 1 then only short term values of the features are needed and used in calculations. If weights are 0,7 (Short) and 0,3 (Long) then tool uses following formulas to count Total Min and Max value.

Total Min Value = 0,7 * Short value min + 0,3 * Long value min
Total Max Value = 0,7 * Short value max + 0,3 * Long value max

Total Min and Max values are then used in the calculations.

5.4.3 Variation

All features should have a given max and min value for short-term and long-term values so that the core values can be calculated. Since its takes lots of work to define all four value parameters for every feature the variations functionality was developed for the tool. This means that only SHORT VALUE and LONG VALUE cells needs to filled out. Those cells mean the average values from certain short and long value intervals. After this the variation functionality can be used. The variation is first defined for long-term and short-term values by giving both cells a value between 0 % and 100 %. After this if Use-button is clicked the tool clears all the old values from min and max cells and recalculates them. The formulas used are the following with values showed in Figure 9. (Short-term variation value 20% and Long-term variation value 10%)

Short value min = SHORT VALUE * (100% – 20% / 2)
Short value max = SHORT VALUE * (100% + 20% / 2)
Long value min = LONG VALUE * (100% - 10% / 2)
Long value max = LONG VALUE * (100% + 10% / 2)

After the variation functionality is used all the features should have min and max variables defined. If some of the feature's value interval has to be specified, new max and min values can be entered manually over the formula in the cell. However each time when variation functionality is used the formula is set again in the cell. If all the min and max values have to be cleared it can be done by clicking the Clear button.

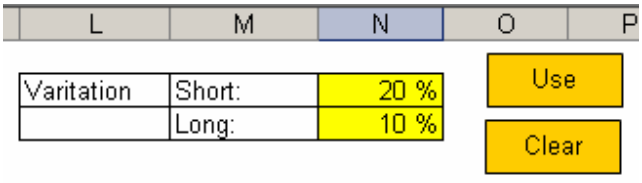

Figure 9. Variation

5.4.4 Features

All the possible features that could be used in a new product release have to be entered in the data sheet. Those features are given a name and a number and their cost and value is defined. There is couple of ways to define the feature values. All the min and max values can be entered manually for long-term and short-term value cells. This way the SHORT VALUE and LONG VALUE cells which indicate the average values of the intervals do not need any values. The other way is to fill out SHORT VALUE and LONG VALUE cells and use variation functionality to calculate min and max values. If the short value weight is defined as 1 then only the short side values need to be entered.

5.5 Defining the dependencies

5.5.1 Mandatory

In Figure 10 is showed how mandatory dependencies between features are defined. First the Mandatory sheet is clicked open. Then for every dependency between features one condition row should be filled out in following way. If selection of feature FN1 requires another feature FN7 to be selected then feature FN1 is given value 1 and FN7 is given value -1. This interdependency is asymmetric which means that selecting FN7 doesn't need FN1 to be selected. Making this interdependency symmetric can be done using two conditions.

There is also a special case when a feature requires more than one other feature to be selected. In Figure 10 condition 3 indicates that feature FN4 requires two features FN9 and FN11 so that it could be selected. In this case the feature that requires more than one feature to be selected is given a positive number which is the same as number of required features. All the required features are given value -1.

	FN1	FN2	FN3	FN4	FN5	FN6	FN7	FN8	FN9	FN10	FN11	FN12	FN13	FN14	FN15
Condition 1			1				-1								
Condition 2		1				-1									
Condition 3				2					-1		-1				
Condition 4															
Condition 5															

Figure 10. Mandatory dependencies

5.5.2 Mutually exclusive

This dependency means that if certain feature is selected then some feature(s) cannot be selected. Figure 11 the condition 1 indicates that either feature FN2 or FN3 could be selected but both of these features can't be in the same product release. In mutually exclusive dependencies there are also some special cases. If more than two features are given value 1 in a row it means that only one feature from this feature group could be selected (Condition 2). With condition 3 and 4 is defined that FN6 could be used only if FN1 and FN10 are not selected but FN1 and FN10 could be used in a same release. One condition could also be that three out of five features could be used in same release. Then all the five features could be given value 1/3 in a row and so only three of them could be chosen for the same release.

	FN1	FN2	FN3	FN4	FN5	FN6	FN7	FN8	FN9	FN10	FN11	FN12	FN13	FN14	FN15
Condition 1		1			1										
Condition 2			1				1	1							
Condition 3	1					1									
Condition 4						1				1					
Condition 5															

Figure 11. Mutually exclusive dependencies

5.5.3 Dummy variables

Dummy variables are used when two features amplify/attenuate each other creating greater/lesser value together than just summing up them individually. Dummy variables are defined on a data sheet below the features and those can have costs and values in the same way as features. The costs of the dummy variables is usually zero and the value can

be positive or negative depending if the dummy is used to amplify or attenuate feature selections. After the dummies are defined they can be used on a Dummy Variables sheet.

The use of a dummy is defined with a two conditions. For example if FN89 and FN96 in the Figure 12 are both selected for the same release then dummy 1 is also selected which either amplifies or attenuates the features. The condition rows have to have then the following values. In the first condition row the two features that affect for the use of dummy have values 1 and the dummy has value -1. On a second condition row the two features that affect for the use of dummy have values -1 and the dummy value is 2 on this row. This way the Dummy is included in a product release only if both of the required features are included too.

FN89	FN90	FN91	FN92	FN93	FN94	FN95	FN96	FN97	FN98	FN99	FN100	Dummy 1	Dummy 2	Dummy 3	Dummy 4
1							1					-1			
-1							-1					2			
				1		1							-1		
				-1		-1							2		

Figure 12. Dummy variables

5.6 Getting results

5.6.1 Corevalues

When all the features and dependencies are inputted in the tool the Corevalues can be calculated using the "Solve Problem"- button that starts solving the linear problem with Excel solver. It could take many hours from the solver to find the dominated portfolios and define the core values. If needed, the process can be aborted using Ctrl + Break and ending the task. After solving the problem the new core values are drawn in a graph when the "Show Corevalues"- button is used. In the chapter 5.7.2 are the corevalues analyzed more precisely.

5.6.2 Dominating portfolios

On a sheet "Portfolios" the dominating portfolios could be viewed. The features that are included in one portfolio are indicated with binary value 1 and features that are not included in a portfolio have binary value 0.

5.7 Testing the tool

5.7.1 Input data

For testing the tool we needed test data. Because we could not gather real data we had to generate our own input for the tool. The test data consisted of 100 different features each having cost, minimum and maximum values.

Test data was generated as follows. The costs were selected from uniformly random distribution between 5 and 25. Maximum value has a negative correlation to the cost with factor -0.5 and variance 2.2. Minimum value is uniformly distributed from 80% to 99% of maximum value. Distribution of values as a function of cost is presented in the picture below.

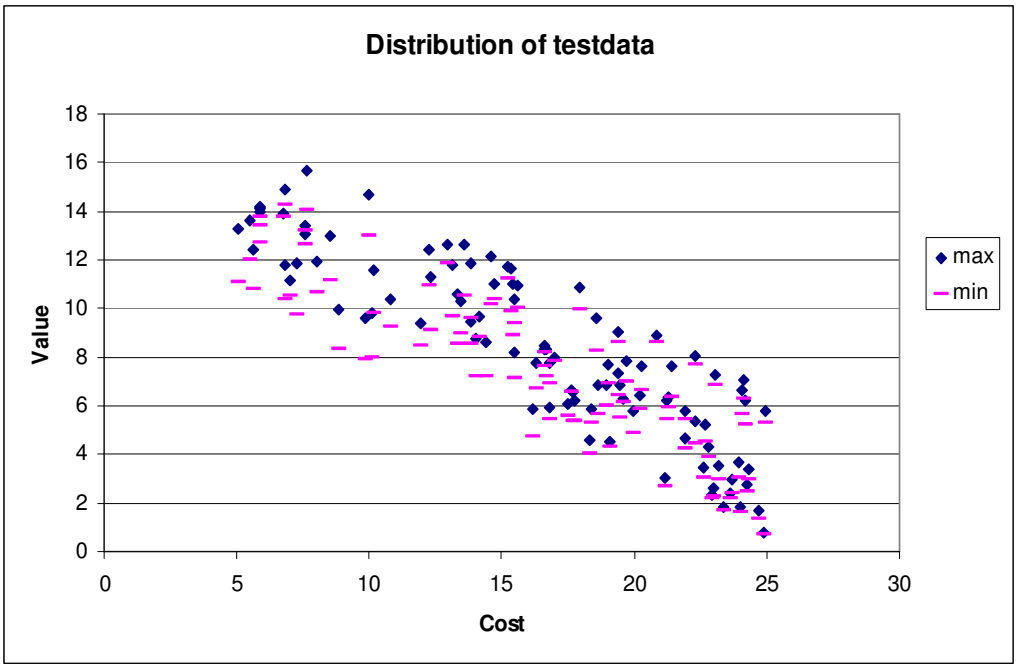

Figure 13. Distribution of testdata

The data has interesting property. It is polarized: there is few very good features (low cost and high value), and there is few very bad features (high cost and low value). When the budget constraint is set correctly algorithm has to include some features from between and make decisions.

In addition to the feature values, some interdependencies were added to the model. There were 9 mandatory, 3 mutually exclusive conditions and 3 dummy projects (creating 6 dummy conditions). The budget of product release was set to 160.

5.7.2 Test results

We run three tests with different software to find out the dominating portfolios. The first test was to use previously developed Excel tool with regular Excel Solver. As a second test we used Matlab's Monte Carlo based algorithm. Third test was to use the same tool as in the first test, but equipped with third party solver.

In the first test the solver found all dominating portfolios from the first ten candidates. The solver was stopped before it reached the maxmin boundary, because the chances fall as the maximum value gets smaller. Monte Carlo method found all the dominating once in the first eight minutes. Third party solver found the last dominating portfolio from the 55^{th} candidate, thus the solver was stopped after about 60 candidates of non-dominating portfolios.

Results from the tests are summarized in the table below.

Table 2. Summarized results from the tests run.

Method	Time	Total portfolios calc.	Dominating portfolios
Regular excel solver	38 min	30	6
Matlab /w MC	16 min	10 000	15
Third party excel solver	16 min	114	25

After running the tests it is interesting to compare the results. Because of the method Excel uses there is always little uncertainty whether all dominating portfolios are really dominating. This error may result from the rounding or the fact that there may be two

different portfolios with same maximum value. If this happens later calculated portfolios may be dominated.

Collecting all portfolios into same table and making pairwise dominance test we get a good hunch of whether all portfolios are dominating or not. Results were that in the first test there were actually 5 dominating portfolios. All second tests dominating portfolios proved to be dominating, which is a natural property of method. In the third test 23 portfolios were dominating ones. After all results were combined 26 different portfolios were found.

When comparing the corevalues (Figure 14 - Figure 16)of the results are almost the same even the number of dominating portfolios is different. In the first test there is just one core feature (FN48) (corevalue 100 %) that proved otherwise. First and third tests include a feature FN41 that was not in any dominating portfolio. Overall the corevalues are very similar with different methods of calculation even the number of dominating portfolios is small. In the Figure 17 are all found dominating portfolios.

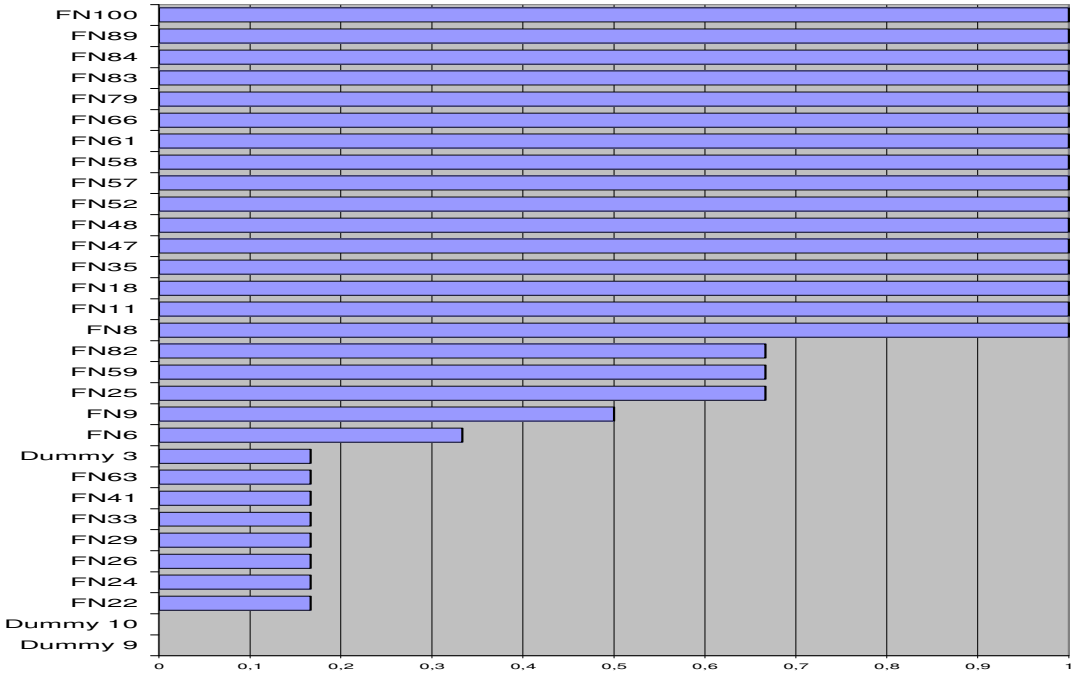

Figure 14. First test corevalues.

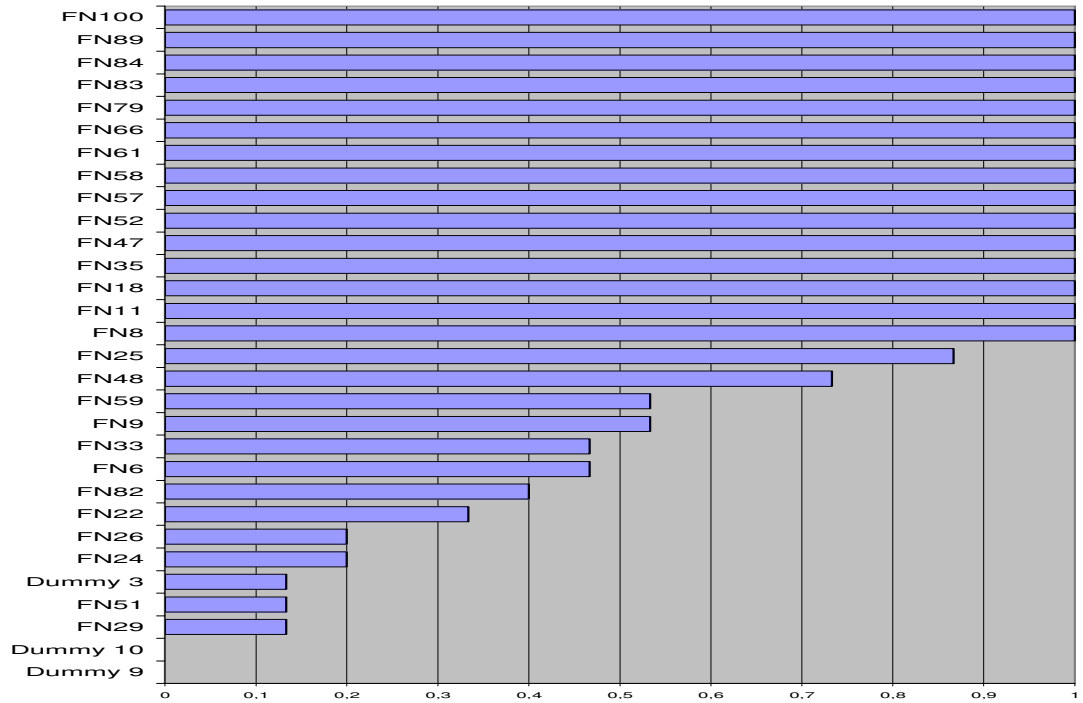

Figure 15. Corevalues from the second test.

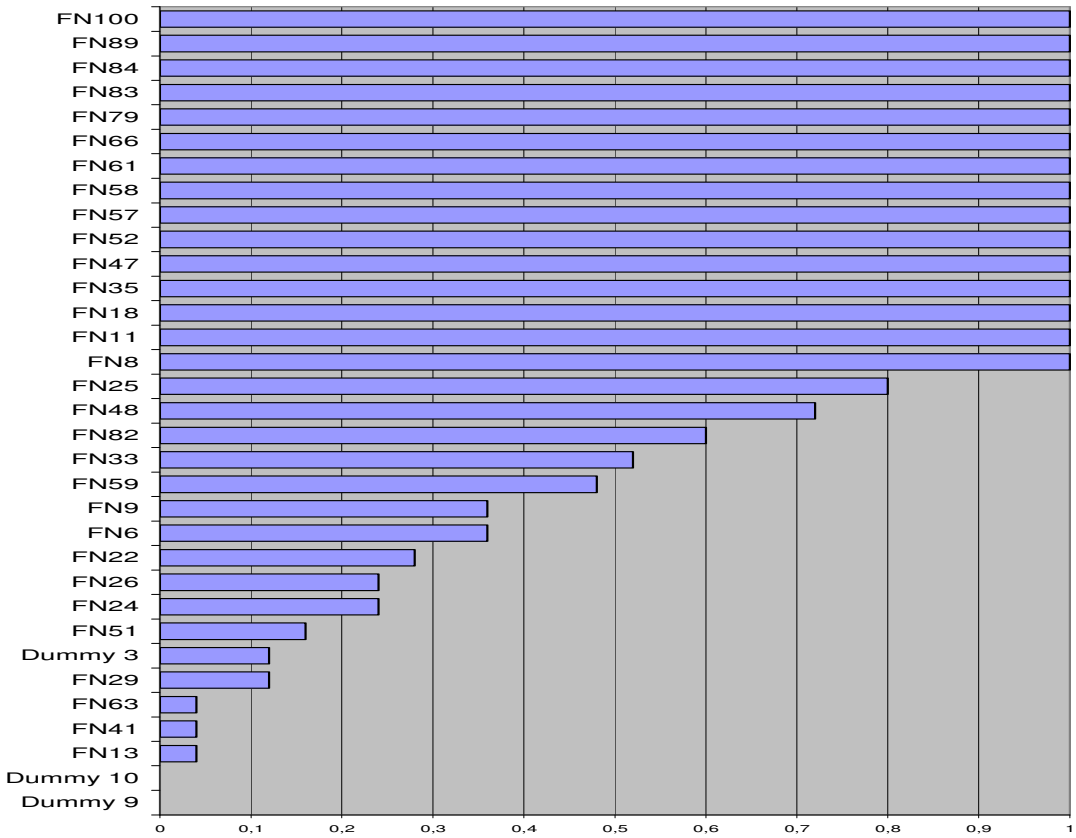

Figure 16. Third test corevalues.

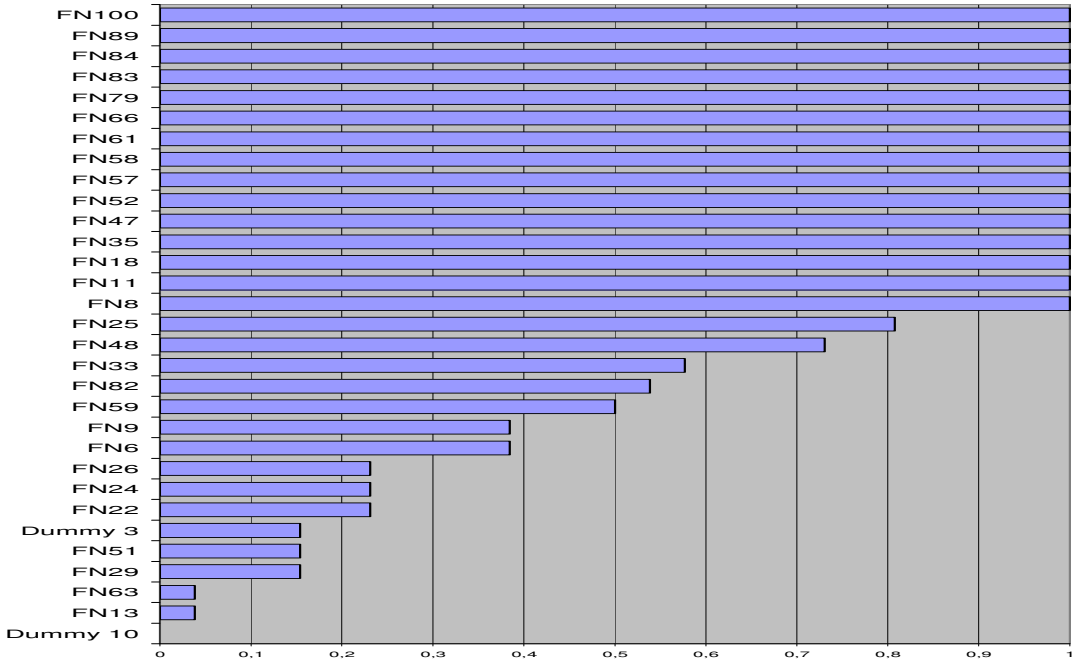

Figure 17. All found corevalues of dominating portfolios.

6 Conclusions and recommendations

6.1 New process model

Based on the literature review a model of the release definition decision process has been developed for the client. It is shown in Figure 18. The release definition decision process for the client. This process model is based on incremental software release planning literature. Each step in the process is divided into necessary activities, shown in Table 3. Activities in the problem definition – step of the release definition decision process.

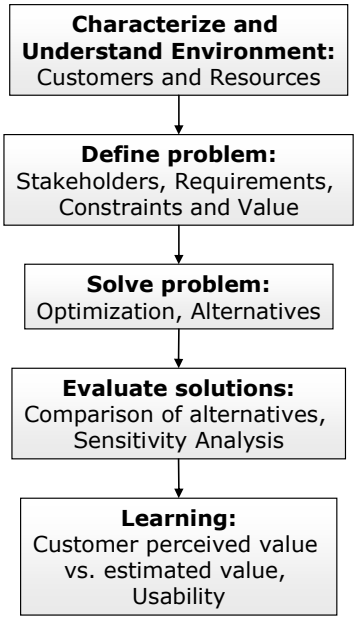

Figure 18. The release definition decision process for the client

In performing the release definition process different approaches described in the literature review can be used. For example in defining requirement and value stakeholders have for these the different methods of requirements engineering and prioritization may be used. In solving the problem a tool is proposed i the following sections, but alternatively the other approaches explained on the literature review may be applied.

Table 3. Activities in the problem definition – step of the release definition decision process

Characterize and understand environment
Identify external factors
demand for the product, customers
Identify resources available
number, skills and availability of people needed

Define problem
Identify the stakeholders:
sales representatives, users, investors, shareholders, project managers, product managers, developers
Elicit and specify the requirements:
information from different stakeholders
business impact of requirements
Identify the relative importance of stakeholders
Define the constraints
precedence, coupling or resource constraints (includes dependencies between requirements)
Stakeholders assign value to requirements
urgency or financial value etc.

Solve problem
Use tool for optimization of requirements

Evaluate solutions
Analyze the set of solutions arrived at
scenario analysis, risk analysis
Add impact of other factors
Forced inclusion or delaying

Learning
Analyze accuracy of stakeholder value assignments
Develop tool based on experience

6.2 Evaluation of the tool

The tool developed in this research proves it is possible to solve portfolio optimization problems with Excel. However, it is evident from testing results that some third party commercial solvers have to be used. The tool is usable, but if it is utilized in wider scale, some further development should be made. Here are some targets for further development:

1. Choosing a dominant portfolio from results could be easier. For example selecting the features from core value sheet would leave out features that are not in dominating portfolios any more.
2. Entering the interdependencies between the features could be simpler. For example selecting two or more features from pull-down menus instead of entering the dependencies to table.
3. Status of the solving process should be viewed better.

There are also alternative choices for the tool. There have been developed more powerful algorithms and software for solving these problems in the system and operation research department of TKK.

7 References

Amandeep, A., Ruhe, G., Stanford, M.: Intelligent Support for Software Release Planning in Bomarius, Iida (Eds.): PROFES 2004, LNCS 3009, pp. 248-262, Springer-Verlag 2004.

Bagnall, A.J., Rayward-Smith, V. J., Whittley, I. M. (2001) The Next Release Problem, Information and Software Technology, 43 (14), pp. 883-890.

Bartlett, W., Raja, M.K. (1995) Application of QFD to the software development process, International Journal of Quality and Reliability Management, vol 12, no 6, pp. 24-42.

Blomqvist M., Hara V., Koivuniemi J. and Äijö T., (2004) "Towards networked R&D management: the R&D approach of Sonera Corporation as an example", R&D management, Vol. 34, No. 5, pp. 591 – 603

Carlshamre, P.: Release Planning in Market-Driven Software Product Development: Provoking an Understanding, Requirements Engineering, 7, 2002, pp. 139-151.

Carlshamre, P., Regnell, B. (2000) Requirements lifecycle management and release planning in market-driven requirements engineering processes, Proceedings of the 11th International Conference on Database amd Expert Syste, Applications DEXA 2000.

Cooper R., Edgett S. and Kleinschmidt E., (2001), "Portfolio management for new product development: results of an industry practices study", R&D management, Vol. 31, No. 4, pp. 361 – 380

Greer, D., Ruhe, G. (2004) Software release planning: an evolutionary and iterative approach, Information and Software Technology, 46, pp. 243-253.

Höst, M., Regnell, B., Natt, J., Nedstam, J., Nyberg, J. (2001) Exploring bottlenecks in market-driven requirements management processes with discrete event simulation, The Journal of Systems and Software, 59, pp. 323-332.

Jin Z. (2001), "The nature of NPD and role flexibility of R&D / marketing in a fast growing high-tech setting", R&D management, Vol. 31, No. 3, pp. 275 – 285

Jung, H.-W. (1998) Optimizing Value and Cost in Requirements Analysis, IEEE Software, pp. 74-78.

Karlsson, J., Wohlin, C., Regell, B. (1998) An evaluation of methods for prioritizing software requirements, Information and Software Technology, 39 (1998), pp. 939-947.

Kauppinen, M. (2005) Introducing requirements engineering into product development: towards systematic user requirements specification (Doctoral dissertation), Helsinki University of Technology, Department of computer science and engineering.

Kulak D. and Guiney E. (2000) "Use Cases: Requirements in Context", First edition, ACM Press, New York, Addison-Wesley, 370 pages

Lee, M., Pham, H., Zhang, X. (1999) A methodology for priority setting with application to software development process, European Journal of Operational Research, 118, pp-375-389.

Liesiö, J., Mild P., and Salo, A. (2005) Preference Programing for Robust Portfolio Modeling and Project Selection, European Journal of Operational Research (to appear).

Momoh, J.A. (2004) Applying intelligent decision support to determine operational feasibility of strategic software release planning (Thesis), University of Calgary, Department of electrical and computer engineering.

Ruhe, G, Greer, D. (2003) Quantitative studies in software release planning under risk and resource constraints, Proceedings of the 2003 International Symposium on Empirical Software Engineering (ISESE'03).

Ruhe, G., Saliu, M.O.: The Art and Science of Software Release Planning, IEEE Software, November/December 2005, pp. 47-53.

Saliu, O., Ruhe, G. (2005) Supporting software release planning decisions for evolving systems, Proceedings of the 2005 29th annual IEEE/NASA Software Engineering Workshop.

Salo, A., Käkölä, T. (2005) Groupware support for requirements management in new product development, Journal or organizational computing and electronic commerce, 15(4), pp. 253-284.

Wang Q. and Montaguti E. (2002), "The R&D-marketing interface and new product entry strategy", Marketing Intelligence & Planning, Vol. 20, No. 2, pp. 82 – 85

Wiegers, K. (1999) First things first: prioritizing requirements,, Software Development, September, 1999.