Determination of the Next Release of a Software Product: an Approach using Integer Linear Programming*

J.M. van den Akker, S. Brinkkemper, G. Diepen, J. Versendaal

Institute for Information and Computing Sciences,
Universiteit Utrecht, P.O. Box 80089, 3508 TB Utrecht, The Netherlands.
e-mail: {j.m.vandenakker, s.brinkkemper, g.diepen, j.versendaal}@cs.uu.nl

Abstract. Selection of the requirements for the next release of a software product is a inherently complex task due to the high volume of intricate requirements and to the varied interests of the stake holders involved. In this paper we apply integer linear programming techniques to aid requirements managers of product software companies in release planning. The applied techniques take candidate requirements, estimated revenue per requirement (or combination of requirements), and available resources as input. Planning suppleness is added by way of allowing flexibility in team composition, team transfers, extension of deadlines and hiring external resources. Through experiments the application of the proposed approach is demonstrated with real life data. Future additions to the model are identified, as well as improved validation.

1 Planning the next release

Companies developing product or embedded software face challenges in determining the set of requirements for an upcoming release. Confronted with hundreds, if not thousands, of requirements proposed by users, potential customers, marketing and sales managers, architects and software engineers, it is far from trivial to select those requirements that add most value to the product. Different stake holders in the planning process may each have their own preferred content in view of the projected revenue estimations and required development efforts. In Figure 1 the requirements management process is depicted in relation to the development of a release. The picture is derived from Natt och Dag et al. (2005) and shows the practice Baan (an enterprise resource planning software company, now part SSA Global). We use this process to describe the context of the release planning process.

Market requirements represent the wishes from existing and potential customers of the software product. Product requirements are defined as generic customer wishes that can be included in the software solution to be developed

* Supported by BSIK grant 03018 (BRICKS: Basic Research in Informatics for Creating the knowledge Society)

Proceedings of the CAiSE'05 Forum - O. Belo, J. Eder, J. Falcão e Cunha, O. Pastor (Eds.)

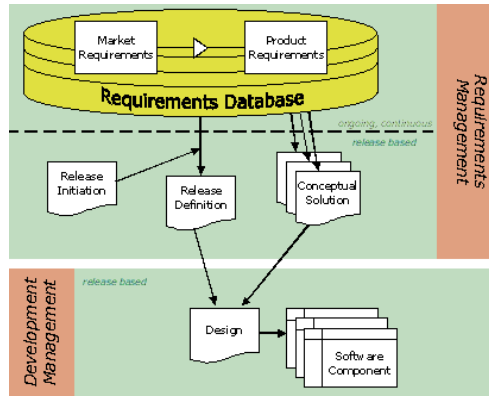

Fig. 1. The requirements management process related to development management

by the software vendor. Release initiation is a formal document triggering the release of a development project. The release definition contains the list of product requirements to be developed in the upcoming release. Conceptual solutions in requirements management are used to sketch the business context of one or more product requirements, and help developers to view the 'big picture' of the development of a product requirement. Finally, the selected product requirements will be developed into software components.

Provided the set of all product requirements, many aspects influence the definition of an optimal set of requirements for a next release. For example, the importance or business value of the requirement, personal preference of certain customers and other stake holders, penalty if not developed, cost of development in man-days, development lead-time, requirement volatility, requirement dependencies, ability to reuse, and requirements quality (e.g. Berander and Andrews (2005); Firesmith (2004); Ruhe et al. (2003); Natt och Dag et al. (2004); Natt och Dag et al. (2005)). So we address what an adequate set of product requirements is for an upcoming release of the software, in the context of business continuity of the product software company.

We develop and demonstrate an optimization technique, based on Integer Linear Programming (ILP) (see e.g. Wolsey (1998) for a general introduction on ILP; see e.g. Van den Akker, et al. (1999A) and Van den Akker, et al. (1999B) for application of ILP in the area of scheduling), to support software vendors in determining the next release of product software. Our technique is based on the assumption that a release's best set of requirements is the set that results maximum projected revenue against available resources in a given time period. We take into account practical aspects, including the total list of requirements, whether or not requirements are dependent on one another, a requirement's projected revenue, and a requirements resource claim per development team. To further increase its application in practice, we enhance our technique with managerial steering mechanisms, i.e. enabling of team transfers, conceding release

deadline extension, allowing extra resources, and more. By introducing these aspects and managerial steering mechanisms into ILP models for the release composition process we extend the work of Jung (1998), who also applied linear programming techniques in the context of release planning.

Our approach is original in three ways. Firstly, we include a unique set of aspects, among others needed team capacity per requirement. Secondly, we introduce unique yet practical managerial steering mechanisms that can help product managers and development project managers in release planning: enabling of team transfers, and deadlines and extra resources. Thirdly, we apply ILP techniques in a unique way.

2 Integer linear programming models for release planning

Let $\{R_1, R_2, \ldots, R_n\}$ be the set of candidate requirements. For each requirement R_j we assume we can estimate its revenue v_j. Further assume that we have a fixed development period. We have to find the subset of requirements for the next release such that the revenue is maximal and the available capacity is not exceeded. We denote our development period by T and define $d(T)$ as the number of working days in the development period. Moreover, let Q be the number of persons working in the development teams of the company. Then, the available capacity equals $d(T)Q$ man-days. Moreover, we have an estimate a_j of the amount of man days needed for the implementation of requirement R_j. We model the requirements selection problem in terms of binary variables x_j $(j = 1, \ldots, n)$, where

$$x_j = \begin{cases} 1 \text{ if requirement } R_j \text{ is selected;} \\ 0 \text{ otherwise.} \end{cases}$$

The problem can be modeled as an ILP problem in the following way:

$$\max \sum_{j=1}^{n} v_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_j x_j \leq d(T)Q, \tag{1}$$

$$x_j \in \{0, 1\}, \qquad \text{for } j = 1, \ldots, n.$$

If the company decides that, in any case, some of the requirements have to be included in the new release, this can be achieved by fixing the corresponding variables x_j at 1, i.e. adding the equation $x_j = 1$ to the above model. If the number of working days in the planning period is different for different persons the total capacity is given by $\sum d_p(T)$, where $d_p(T)$ is the number of working days of person p in period T and the sum is over all persons in the company.

Especially in larger product software companies there are different development teams, each with their own specialization. Let m be the number of teams

and suppose team G_i $(i = 1, \ldots, m)$ consists of Q_i persons. We assume that the implementation of requirement R_j needs a given amount a_{ij} of man days from team G_i $(i = 1, \ldots, m)$. Now the team capacities can be included by replacing constraint 1 by:

$$\sum_{j=1}^{n} a_{ij} x_j \leq d(T) Q_i, \quad \text{for } i = 1, \ldots, m. \tag{2}$$

Note that for $m = 1$, this coincides with the model for one pool of developers.

By allowing people to work in a different team, there is more flexibility that can increase revenue. We assume that if a person from team G_i works in another team G_k his contribution in terms of man days is multiplied by α_{ik}, i.e. if the person works one day this contributes only α_{ik} to the work delivered by team G_k. Besides the variables x_j, we now define the variables y_{ik} as the number man-days from team G_i deployed in team G_k. Let m_i be the number of man-days in team G_i. Including transfers results in the following model:

$$\max \sum_{j=1}^{n} v_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \leq y_{ii} + \sum_{k:k \neq i} \alpha_{ki} y_{ki} \qquad \text{for }, i = 1, \ldots, m, \tag{3}$$

$$\sum_{k=1}^{m} y_{ik} = m_i, \qquad \text{for } i = 1, \ldots, m, \tag{4}$$

$$x_j \in \{0, 1\}, \qquad \text{for } j = 1, \ldots, n,$$

$$y_{ik} \text{ nonnegative and integral}, \qquad \text{for } j = 1, \ldots, n.$$

3 Experimentation

As a proof-of-concept we implemented a prototype of a requirements selection system. For testing the program different data sets were used. The different sets were: with 9 requirements and 3 teams, with 24 requirements and 17 teams and different ratios of available and total required capacity and with 99 requirements and 17 teams. All of the used data sets are available online for research purposes at `http://www.cs.uu.nl/~diepen/ReqMan`.

The following table presents the total revenues for the optimal requirement selection for the different problems that were tested.

Data	Pool	No-X-Teams	X-Teams
small	182	147	182
AA	700	510	685
BB	810	570	790
CC	835	670	810
master	46220	42730	44810

Table 1. Revenue values for the three types of data sets

The first column in Table 1 represents the scenario of a development organization consisting of one big pool of developers. In practice, however, there will be different teams (especially with larger development organizations); the second column therefore depicts the scenario with multiple teams in the development organization (with no transfers allowed). The last column depicts the scenario with multiple teams, and team transfers allowed ($\alpha_{ik} = 0.7$ for all $i \neq k$). The case with one big pool of developers leads to the highest revenue. Introducing multiple teams will decrease total revenue. From the theory this is explained by the fact that this model imposes stricter conditions in assigning labor to requirements. However, allowing transfers between teams (for $\alpha_{ik} = 0.7$ for all $i \neq k$) again increases flexibility and hence revenue.

4 Extensions to the base model

Until now we have assumed that all requirements can be implemented independently Suppose that the functionality imposed by a certain requirement can only be effective if one or more other requirements are also implemented, say if we select requirement R_j we also have to select R_k. In the model, we have to ensure that

$$x_j = 1 \Rightarrow x_k = 1.$$

This can be done by extending our model with the linear inequality

$$x_j \leq x_k. \tag{5}$$

Note that if two requirements are dependent in the sense that they cannot be realized separately from each other, this can be handled by considering them as one requirement.

Until now, we assumed that the total revenue of a release is the sum of the revenues of the individual requirements included in the release. However, the revenue may be larger if some collection of requirements is realized. A typical example is when these requirements together provide a 'package' in some area within the software product. We can include this in our model by using a binary variable x_S indicating if the set S is completely realized. Further details are omitted for reasons of brevity.

To increase the capacity for the development of the next release, the company may consider to hire external personnel in some of the development teams. In our model, this can be included by to adding the external capacity to the right-hand side of constraints (2) and including the cost in the revenue function. Again, we omit further details for reasons of brevity. Another possibility is postponing the delivery date for the new release. This can be included in a similar way.

5 Conclusions and future research

We have shown that ILP models and methods can be used in flexible release planning by implementing a subset of the managerial steering mechanisms in a

prototype tool. ILP techniques can be used to help product and requirements managers in software release planning. In this paper we specifically optimized revenues against available resources in a given time period. Scenarios dealing with one pool of developers, different development teams, including the possibility of transfer of developers between teams are discussed and demonstrated through experimentation. Moreover a number of extensions were modeled.

Our approach simplified the task of computing the set of release requirements to a great extend. We aim for more support during release development as in the dynamics of the project workload turns out to be overestimated or underestimated, and that the revenue projections are changing due to a changing market. We further plan to test the validity of our model through multiple business cases. In such cases an appropriate approach is to use our model for release planning on the one hand, and compare its outcome with the company's proprietary way of release planning. We finally note that our approach supports the release planning at the start of the release planning process. In practice the value of requirements may evolve over time, as the release is being developed; dealing with this is a further extension of our current approach.

References

1. van den Akker, J.M., C.P.M. van Hoesel, and M.W.P. Savelsbergh (1999). A polyhedral approach to single-machine scheduling problems. *Mathematical Programming 85*, 541-572.
2. van den Akker, J.M., J.A. Hoogeveen, and S.L. van de Velde (1999). Parallel machine scheduling by column generation. *Operations Research*, Vol. 47, No. 6, 862-872.
3. Berander, P. and Andrews, A. (2005), Requirements Prioritization. In: *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin (eds.), Berlin, Germany, Springer Verlag (Forthcoming).
4. Jung, H.-W. (1998), Optimizing Value and Cost in Requirements Analysis, *IEEE Software*, July/August 1998 pp 74-78.
5. Natt och Dag, J., Gervasi, V., Brinkkemper, S. and Regnell, B. (2005), A Linguistic Engineering Approach to Large-Scale Requirements Management, *IEEE Software, Special Issue on Requirements Engineering*, vol 22, no 1, pp 32-39, January/February 2005.
6. Natt och Dag, J., Gervasi, V., Brinkkemper, S. and Regnell, B. (2004), Speeding up Requirements Management in a Product Software Company: Linking Customer Wishes to Product Requirements through Linguistic Engineering. *In: Proceedings of the 12th International Requirements Engineering Conference*, N.A.M. Maiden (Ed.), IEEE Computer Science Press, pp 283-294, September 2004.
7. Ruhe, G., Eberlein, A., Pfahl, D. (2003), Trade-off Analysis for Requirements Selection, *International Journal of Software Engineering and Knowledge Engineering,* vol 13, no 4, pp 345-366.
8. Wolsey L.A. (1998) *Integer programming* Wiley-Interscience Series In Discrete Mathematics and Optimization.