Requirements Eng (2010) 15:337–358
DOI 10.1007/s00766-010-0097-5

ORIGINAL RESEARCH

When-to-release decisions for features with time-dependent value functions

Jim Mc Elroy · Guenther Ruhe

Received: 29 April 2008 / Accepted: 19 January 2010 / Published online: 16 February 2010

Abstract Release planning is of key importance for incremental software product development. With the increasing size and complexity of software products, as well as with the growing demands for transparency and objectivity of decision-making, intuition alone is no longer sufficient for release planning. EVOLVE$^+$ is a systematic method for release planning, which combines the strengths of formalization and computational efficiency with the expertise of the human experts. From a given set of candidate features, the most attractive ones are selected and assigned to a sequence of releases. In this paper, we consider three extensions to the current systematic planning approach EVOLVE$^+$: (a) value functions describing the estimated value of the feature are continuous functions of time, (b) the actual release dates are no longer fixed but can be varied in some pre-defined interval. As a consequence, the available resource capacities are also functions of time. (c) Calculation of trade-off solutions balancing the risk with the potential additional value of early release. All three extensions substantially increase the complexity of the release planning problem. We have developed a solution method using genetic algorithms that is able to accommodate the additional complexity of the advanced model. A hypothetical case study is conducted as a proof-of-concept for the applicability of the method. The applicability of the method is demonstrated by analyzing six detailed planning scenarios.

Keywords Release planning · Value-based planning · When-to-release decisions · Time-dependent value · Genetic algorithms · Trade-off solutions · Requirements engineering decision support

J. Mc Elroy · G. Ruhe (✉)
Laboratory for Software Engineering Decision Support,
University of Calgary, 2500 University Dr. NW,
Calgary, AB T2N 1N4, Canada
e-mail: ruhe@ucalgary.ca

J. Mc Elroy
e-mail: jrmcelro@ucalgary.ca

1 Introduction

It has been widely recognized for many years that iterative and incremental development of software offers many benefits over non-incremental methods, as discussed in [1, 6, 8, 15, 20], among others. Planning approaches for iterative and incremental development, known as release planning, arrived later on the scene. These release planning approaches vary from informal approaches, as used in agile "planning games" [12] to more rigorous and formal approaches [31] provides a comparative analysis of seven of the more prominent approaches.

With the exceptions of [12, 13, 21], most release planning methods do not consider time-dependent changes to the value of features in determining which features go into which releases. Denne and Cleleand-Huang [13] as well as Maurice et al. [21] use a single value function—Net Present Value (NPV)—in determining how the release of features affects the income potential of a software development organization. Cohn [12] discusses several economic considerations, but in the context of informal agile release planning. Van den Akker also states that revenue should be the primary denominator for value of features, but does not elaborate on this [32].

However, value considerations may affect release planning, and a method of formally handling any type of value function needs to be explored. In the real world, different factors may influence the value of a feature at different points in time. This paper takes the position that the value of software should be evaluated by financial experts, and that the value figures should be stated in the ratio scale of net income (also called revenue). This may require mapping by financial experts between stakeholder value systems and economic value systems, a topic needing further research. However, for the rest of this paper, we will assume that value can be stated in terms of (estimated) net income.

While providing a potentially more accurate model for release planning, expressing feature values and resource capacities as functions of time also allows for the exploration of the effects of varying release dates, providing a more flexible handling of release planning. The main contributions of the paper are:

(1) Formulation of time-dependent release planning where time-dependent functions express both the value of features and the resource capacities available for implementing features,
(2) A solution method using genetic algorithms to determine trade-off release plans allowing variation of the release dates and/or resource capacities within pre-defined intervals of feasibility in conjunction with a defined measure of risk, and
(3) Providing a proof-of-concept for the proposed approach by running a case study and analyzing the results of the case study for six related planning scenarios.

The paper is subdivided into seven sections. Section 2 explores related work. Section 3 provides the problem statement and related results. The solution approach is presented in Sect. 4. The benchmark case study comes from the telecommunication domain and is studied in Sect. 5. Applicability of the method is discussed in Sect. 6. Finally, summary and conclusions are given in Sect. 7.

2 Related work

Requirements engineering is a decision-centric discipline [2]. As part of that, release planning is an early stage effort to select and assign features to one or more releases in a way that customer satisfaction is maximized. Release planning is cognitively and computationally difficult to solve. This research is part of a larger effort to provide decision support in requirements engineering as outlined in [26].

Jung first proposed a cost-value requirements analysis using mathematical programming [18]. This was an improvement of the work of [19] who proposed the use of the Analytical Hierarchy Process [30] and subsequent intuitive inspection of what constitutes the most valuable requirements. Instead of relying on intuition alone, Jung applied a rigorous algorithm for solving the resulting knapsack problem to decide which requirements should be placed in the next release.

van den Akker et al. [32] applied mathematical programming to provide a solution for the next release problem. In contrast to other methods, their model takes into account dependencies between requirements, additional revenue for bundles of requirements, and a requirement resource claim per development team. To further increase practical applicability, different resource usage and transfer modes, taking into account different aspects and managerial steering mechanisms, are used.

More recently, advanced optimization-based support systems have been created. The EVOLVE$^+$ method [25] is an extension of the former EVOLVE* approach [29]. It strictly follows the evolutionary problem solving paradigm and introduces a set of qualified alternative solutions at all iterations. This capability is used to facilitate the handling of implicit concerns in addition to the explicitly defined constraints. The key assumption for this method is that structural diversity of a set of qualified release plan solutions gives a better chance to solve the right problem.

The EVOLVE$^+$ method has been implemented in a web-based decision support system called ReleasePlanner^{TM} [28]. The technology has proven successful in numerous real-world projects. Practical experience has been reported in [1, 5, 23]. While the granularity of release points is flexible, the purpose of planning in EVOLVE$^+$ is more strategic than operational. What that means is that resources are considered cumulatively over the chosen release (or iterations) interval. In subsequent operational planning, the follow-up question of deciding who is doing what at which point in time is decided (see [27]).

EVOLVE$^+$ is based on a number of assumptions which are not necessarily fulfilled in all cases. We will mention three limitations which are addressed in the method proposed in this paper. The first assumption is about fixed release dates. Most of the current methods for release planning assume that release dates are defined up front. While this is often reasonable, in reality, there is often the need to adjust the dates based on additional considerations such as quality. Bhawnani et al. [4] provide a method to address the when-to-release question based on the estimated reliability of the release product. The proposed approach evaluates the suitability of existing reliability models in guiding when-to-release decisions.

The trade-off between product performance and time to market was examined in [11]. The authors showed how the optimal time to market is affected by the size of the potential market, the presence of existing products, the size of the profit margins, the length of the opportunity window, the speed of product performance improvement, and the performance level of competitive products.

The second assumption relates to the way in which the value of a feature is taken into account for planning. The value is expressed as a (weighted) stakeholder scoring average using a nine-point scale. In addition to that, all feature values are simultaneously downgraded in releases past the current planned release. For all features, there is the same value ratio when considering different options for release.

Three case studies for creating software product value through requirements selection are studied in Barney et al. [3]. The authors concluded that "… creation of software product value is not well understood, it cannot be managed in the most effective way" [3]. In the context of planning for value, Cohn [12] discusses four types of potential value (revenue) for an item being (a) new revenue, which represents new revenue streams opened up by a released item, (b) incremental revenue, which refers to additional revenue obtainable from existing customers, (c) retained value, which represents revenue that will not be lost from existing customers, and (d) operational efficiencies. Denne and Cleland-Huang [13] have a slightly different list of potential revenue types, which adds the following (somewhat overlapping) types to the above: (e) competitive differentiation, (f) brand-name projection, and (g) enhanced customer loyalty.

Finally, as a third limitation, risk is not explicitly taken into account in EVOLVE$^+$. In addition to value, each plan x has an associated risk Risk (x). When-to-release decisions are impacted by some inherent technical and quality risks. The fundamental trade-off is to balance between an appropriate level of risk for when-to-release and achieving a sufficiently good value that way.

The only way to bring risk into consideration in EVOLVE$^+$ is using the stakeholder scoring process to evaluate a specified risk for the features. However, this neither addresses the implementation or acceptance risk related to provision of features at releases, nor does it reflect any type of time-dependent variation of the risk (in dependence of the release date).

3 Problem statement

3.1 Baseline model EVOLVE$^+$

In what follows, we present the formulation of the EVOLVE$^+$ release planning model [29] that serves as a baseline model for the purposes of this paper. For simplicity, this paper will call all types of prioritizable items considered for release planning "features". A feature is defined in [33] as a logical unit of behavior that is specified by a set of functional and quality requirements.

We assume a set F of N features $F = \{f(1), f(2),\ldots, f(N)\}$. For planning, we need to define how long in advance we want to plan. This is the number of product releases we consider for planning purposes. In general, we assume consideration of the next K releases of the product to be planned for. For our baseline model, the K different release dates called rd(1) …rd(K) are assumed to be fixed.

A release plan is characterized by a vector x of decision variables

$$x = (x(1), x(2), \ldots, x(N)) \text{ with}$$

$$x(i) = k \text{ if feature } f(i) \text{ is assigned to release } k \in \{1, 2, \ldots K\}, \text{ and} \quad (1)$$

$$x(i) = 0 \text{ otherwise.} \quad (2)$$

Our formulation considers dependencies between features in the form of coupling and precedence. Coupling between two features means that they should be offered in the same release. Precedence of a feature $f(i)$ related to a feature $f(j)$ means that $f(j)$ should not be offered in a release earlier than feature $f(i)$.

The estimated consumption of resources by features is considered part of the modeling as well. Different methods are applicable for providing estimates. Most of these methods incorporate some form of learning from former projects. In addition, often these methods are hybrid in the sense that they combine formal techniques with the judgment and expertise of human domain experts. For an overview of resource estimation methods, we refer to [7].

The definition of the actual resource types is project specific and is done by the product manager. The resource units can be defined in a project-specific manner as well. We distinguish between human and non-human resources (such as finances). For human resources, person days or person weeks are most frequently applied as units of measurement. Resources can be defined in correspondence of the product life-cycle role (e.g., design, implementation, testing) or related to specific skills (e.g., user interface implementation, hardware–software co-design, tool ABC) and/or in correspondence to specific skills needed (such as the ones related to tools or languages).

When planning for R different resource types and K releases ahead of time, each proposed product release plan needs to fulfill the R*K-related resource constraints. Per release, in each of the R constraints, the summation is taken over all the features f(n) assigned to the first release. For a given release plan x and for all types r ($r = 1\ldots R$) of resources, the total consumption within release period k ($k = 1\ldots K$) is called Consumption (k, r, x) and defined as the resource consumption of all features assigned to k.

The Capacity in general is defined as the product of the average number of resource units available per time unit multiplied by the duration of the release. Consumption (k, r, x) is not allowed to exceed the respective capacity called Capacity (k, r).

$$\mathrm{Consumption}(k,r,x) = \sum_{n:x(n)=k} \mathrm{consumption}\,(n,r) \le \mathrm{Capacity}(k,r) \quad \text{for } k = 1\ldots K \text{ and } r = 1\ldots R \tag{3}$$

The criteria for release planning are project specific. Often, the objective(s) of planning are related to creating maximum business value, to minimize time-to-market of features or to maximize satisfaction of stakeholders with the proposed plans. The actual utility function used for planning tries to bring the different aspects together in a balanced way.

For the baseline model, we assume a total value function Value(x) expressing the cumulative business value of all features assigned to releases according to plan x. Value(x) is composed of individual values $v(n,k)$ of the features, where $v(n, k)$ describes the expected value contribution of feature $f(n)$ in case it is assigned to release k according to plan x. For the baseline model, we assume point-wise value estimates $v(n, k)$. For further details on the composition of the coefficients $v(n, k)$, we refer to [29].

3.2 Extension 1: time-dependent value of features

In the baseline model, the value of a feature is considered to be a single value point estimate. However, the value of a feature might change over time. In general, it would be expected that the potential market value of an item would drop with time.

In Fig. 1, we illustrate different types of functions which are later used in the case study example. The "present value" of money argues that money currently available is typically worth more than money obtained in the future, because "present money" can be invested (see features $f(1)$ and $f(2)$ in Fig. 1).

Other factors will also cause a drop of the marketable value of an item, e.g., a competitor might release a similar product or a product with a similar feature, or the novelty of a new item would simply deteriorate with time (also reflected in the value curves of features $f(1)$ and $f(2)$ in Fig. 1). If the approximate release date of a competitor's similar product is known, then a significant drop-off in value after the competitor's release date might be expected, especially if the competitor's product is released first (compare feature $f(3)$ in Fig. 1).

Conversely, the value of some items may rise with time. For example, an item, such as a new operating system, may require more computational power than is currently available, but which will become common in a year or two (see feature $f(5)$ in Fig. 1).

Similarly, user acceptance and demand for any new product may experience a similar rise in value over time. Value might also peak near specific dates. For example, the value of new game software may peak around holidays (Feature $f(4)$ in Fig. 1). In addition, marketing plans may call for specific timing on the release of items in the future, although they could be developed and released earlier.

Daily income formulas can be expressed as piece-wise linear functions. "Present value" formulas are often

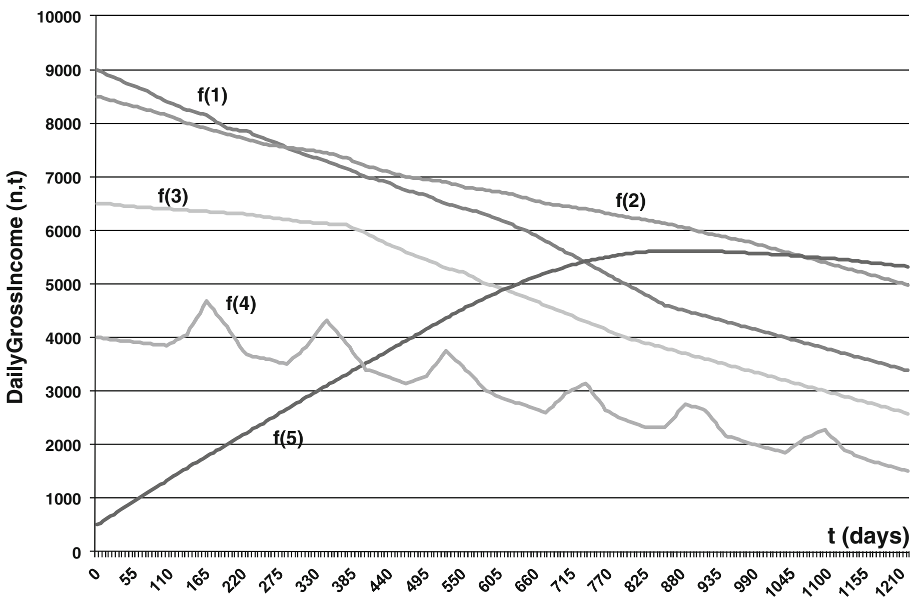

Fig. 1 Time-dependent value functions for market-driven features

applicable, with varying rates of value decline. The earlier work leading to this paper [22] used such formulas.

However, using such formulas raises the questions of how they are derived and who derives them. The current paper uses an approximation approach, where periodic (monthly and/or quarterly) income estimates per feature are agreed upon by appropriate stakeholders, and these are converted to daily income estimates by dividing the periodic estimates by the number of days in the period. The period midpoints of the daily estimates are then plotted to arrive at a series of piecewise linear equations. The curves in Fig. 1 actually are graphs of piecewise linear functions. Due to the relatively short period intervals in comparison with the overall time interval, the combinations of the piecewise linear parts take on the appearance of curves.

Daily income formulas are not applicable for some features. For example, for contract-driven software, income may be the same regardless of the delivery date, as long as deadlines are met, after which point significant penalties may apply. The value functions for these types of software are best expressed in terms of Total Net Value (TNV). TNV functions show what the total net income, from a given time to the time horizon, of a feature would be if that feature is released on a specific date.

Figure 2 shows the TNV of contract-driven features, with steep penalties applied if deadlines are not met. Again, these sample functions stem from the case study example studied in Sect. 5. Note that even without penalties, "constant" values can deteriorate somewhat with time due to present value considerations.

In this paper, we assume that value of features is expressed in total net value. TNV is either directly provided or it can be determined by the integration of daily gross income functions.

We consider T time units of time where T is defined by the release date rd(K). With fixed cost called fcost(n) for feature $f(n)$, the total net value from a given time (release date) t to the time horizon T is denoted by TNV(n, t) and is expressed as:

$$\mathrm{TNV}(n,t) = \int_{t}^{T} \text{daily gross income}\,(n,\, t)\ \mathrm{dt} - \mathrm{fcost}(n) \qquad (4)$$

For the sake of simplicity, this paper assumes that TNV functions represent total revenue functions for features and that summation of functions for different revenue types has already occurred. Figure 3 shows the TNV curves for features $f(1)$ to $f(5)$ for which the daily value functions were plotted in Fig. 1. We observe that TNV(n,t) is a decreasing function for all five features. Moreover, the total net value becomes negative after some point in time.

3.3 Extension 2: variation of release dates

The second extension compared to the baseline model is the flexibility in release dates. Unlike the baseline model, we allow variation of the release dates in some fixed interval. For all releases $k = 1\ldots K$, each release date rd(k) is allowed to vary in some interval [rd1(k), rd2(k)]. Correspondingly, the capacity bounds for all releases and all resource types become time-dependent functions as well.

Capacity (k, r, t) denotes the amount of resources of type r available during release period k at point in time t.

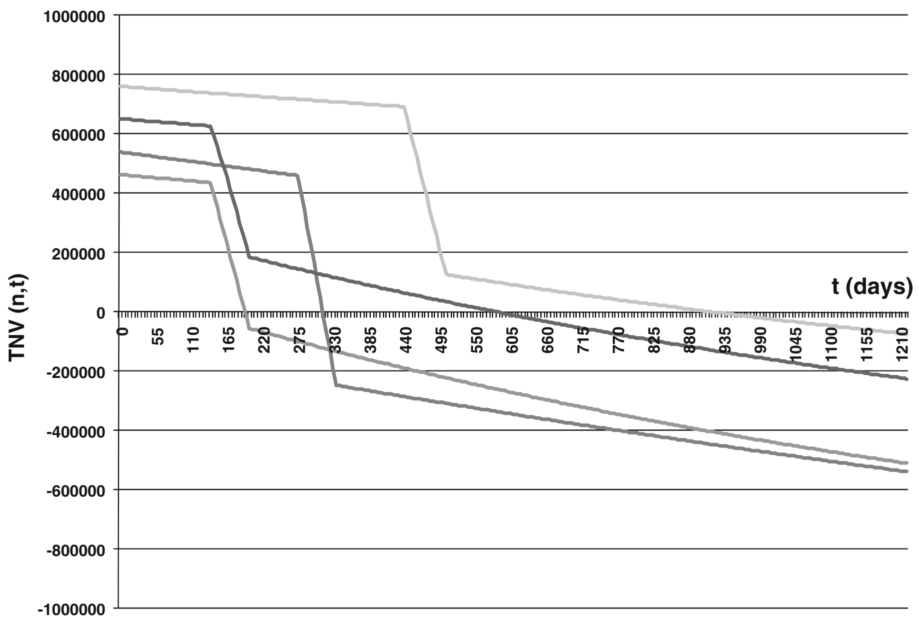

Fig. 2 Time-dependent value functions for contract-driven features

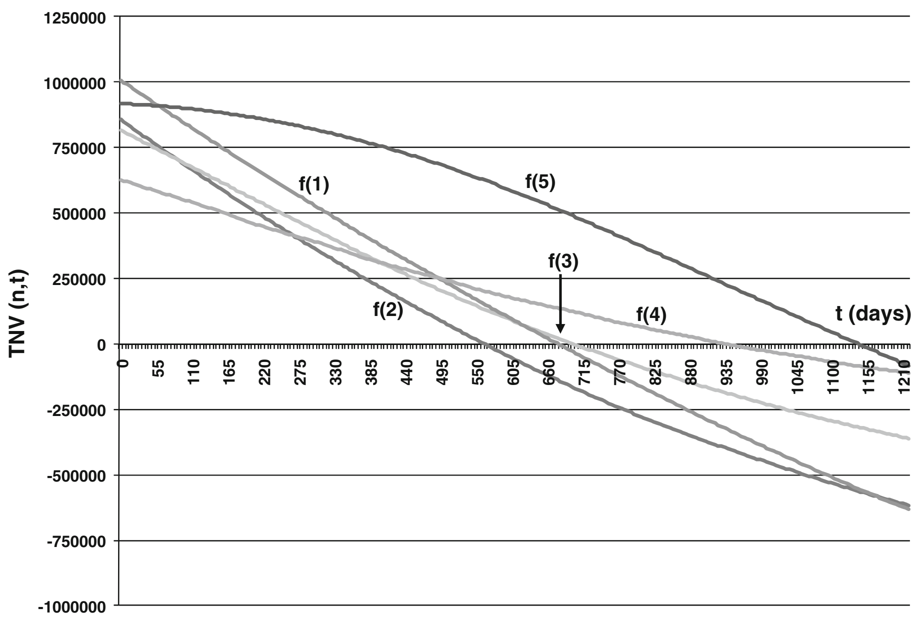

Fig. 3 Total net value functions for features *f*(1) to *f*(5)

For any fixed vector RD of release dates with RD* = ($t^*(1),\ldots, t^*(K)$) and $t^*(k)$ taken from the interval [rd1(k), rd2(k)], the resource constraints now become time-dependent constraints summarized by:

$$\text{Consumption}\,(k,r,x) = \sum_{n:x(n)=k} \text{consumption}\,(n,r) \le \text{Capacity}(k,r,t^*(k)) \quad \text{for } r = 1\ldots R \text{ and } k = 1\ldots K \tag{5}$$

3.4 Extension 3: value-risk trade-off plans

There are several types of risks when proposing a release plan x [16]. Here, we consider the balance between risk and value. The risk measure of a given plan x and a given release date vector RD is expressed as a linear function of the differences between the actual release dates rd(k) and the latest possible release date rd2(k):

$$\text{Risk}(x,\ \text{RD}) = \sum_{k=1\ldots K} \alpha(k)\,[\text{rd2}(k) - rd(k)]^{\beta(k)} \tag{6}$$

Factors $\alpha(k)$ ($k = 1\ldots K$) are used to express the relative importance of the risk contribution in dependence of the release. If $\alpha(1)$ is bigger than $\alpha(2)$ then this means that the risk of being early at release 1 is higher than the risk related to release 2. Analogously, exponents $\beta(k)$ ($k = 1\ldots K$) can be used to express non-linear risk behavior related to the time difference [rd2(k) − rd(k)].

Our definition (6) of the total risk is justified by the assumption that an earlier release time is often achieved by reduced effort on quality assurance, especially testing. So, the “risk of being early” is really the well known quality risk, where quality is loosely defined as a lack of defects. However, the method described in this paper remains applicable for any other measurable risk as well.

3.5 Overall problem statement

For given vector RD of release dates rd(1)…rd(K), the totality of technological and resource constraints is abbreviated by X(RD). For a given time horizon of planning T and our assumed set F of features with time-dependent value functions as defined in (4), we can define the total value obtained from all assigned features as

$$\text{Value}(x,\text{RD},T) = \sum_{k=1\ldots K}\ \sum_{n:x(n)=\text{k}} \text{TNV}(n,\text{rd}(k)) \tag{7}$$

Based on the total value function (7), our Dynamic Value-based Release Planning problem called DVB-RP can be formulated as:

$$\begin{aligned}&\text{Trade-off}\{[\text{Value}(x,\text{RD},T),\text{Risk}(x,\text{RD})]\}\\&\text{subject to}\\&x \in X(\text{RD}), \text{RD} = (\text{rd}(1)\ldots\text{rd}(K)) \text{ with}\\&\text{rd}(k) \in [\text{rd1}(k),\text{rd2}(k)]\,\text{for all } k = 1\ldots k\end{aligned} \tag{8}$$

Looking for trade-off solutions means determining plans x^* such that there is no other plan x’ with

$$(\text{Value}(x',\ \text{RD},\ T),\ \text{Risk}(x',\ \text{RD})) \le (\text{Value}(x^*,\ \text{RD}, T),\ \text{Risk}(x^*,\ \text{RD}))\ \text{and} \tag{9}$$

$$(\text{Value}(x',\ \text{RD},\ T),\ \text{Risk}(x',\ \text{RD})) \ne (\text{Value}(x^*,\ \text{RD},\ T),\ \text{Risk}(x^*,\ \text{RD})) \tag{10}$$

Conditions (9) and (10) together ensure that there is no plan x', which is at least as good as x^* but has not the same values for the two criteria. Solutions fulfilling these conditions are called “non-dominated”. The actual number of non-dominated solutions for a given problem varies in dependence of the concrete problem instance. However, there is more than one non-dominated solution in general. For the case study analyzed in Sect. 5, there are 11 non-dominated solutions the product manager can look at in the first iteration.

The fact that both value and resource capacities change with time presents one of the key challenges of this research. As time progresses, the value of features may decrease, but the capacity for implementing more features increases. Therefore, release planning takes on the dimension of determining the “best time to release”, rather than simply fixing a release date then determining the best set of items to implement before that date within the fixed constraints of the problem. This added dimension has not been considered in any of the former release planning approaches studied.

4 EVOLVE$^+$ for DVB-RP

4.1 Overview

In evolutionary problem solving as suggested by the family of solution algorithms following the EVOLVE method, a suitable solution is developed through evolution due to the interaction between the human expert and the results obtained from running computational algorithms based on a formalized problem description. The same principle is applied here for the advanced formulation DVB-RP addressed in this paper. We call the resulting method EVOLVE$^+$ (DVB-RP).

EVOLVE$^+$(DVB-RP) consists of three main phases, which are iterated over until a final acceptable solution is obtained. This iteration involves modeling the problem space, exploration of this space by using computational intelligence to provide a set of candidate solutions based on the models, and in consolidation allowing human experts to analyze the solution sets and change the model if revised understanding is gained. If the model has been changed, then the process is iterated again.

4.2 Phase 1: modeling

The modeling phase is concerned with the following tasks:

- Considering a set F of features with corresponding decision variables x as defined in (1) and (2).
- Defining or updating the time-dependent value functions (7).
- Defining or updating the time-dependent resource constraints (5).
- Defining or updating the risk function (6).

It is assumed that defining or updating of value and resource estimates, and value and resource functions is accomplished via the consolidated opinion of expert stakeholders. For the value functions themselves, we have assumed that point-wise value estimates could be provided and that the value function is composed as a piece-wise linear function from these estimates. This assumption is more applicable the more mature the underlying development processes and the more stable and predictable the related business context is. All point-wise estimates can be the result of expert judgement, either on an individual or on a collective basis.

4.3 Phase 2: exploration

A formalized description of the problem allows the application of rigorous methods and techniques. There exists a wide range of methods and techniques that have proved themselves useful in very different contexts. The different optimization techniques including exact as well as heuristic approaches fit into this category. Search-based techniques such as the ones based on genetic algorithms or simulated annealing are getting more and more attention because of their success in providing sufficiently good solutions to very complex problems in reasonable time [10, 17].

A genetic algorithm (GA) was developed to solve the value-based time-dependent release problem (8). Chromosomes in the program’s genetic algorithm represent release plans, with gene positions in the chromosomes indicating specific releasable features, and values of the genes representing the release number in which the feature will appear according to that plan (see Fig. 4).

The genes reference contains full feature information, such as resource consumption, precedence and coupling constraints, and resource capacities. This information is used for determining feasibility and overall fitness of each chromosome. Random generation or mating creates chromosomes that are either feasible, fulfilling all constraints, or infeasible. In our implementation, infeasible chromosomes are discarded, while feasible chromosomes are evaluated for their “fitness”.

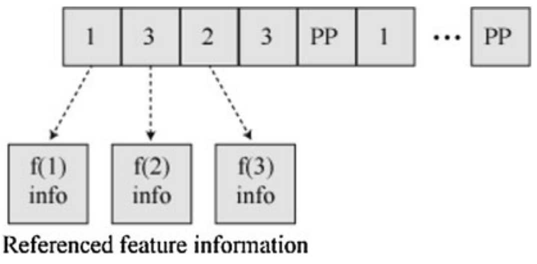

Fig. 4 Chromosome representing a sample release plan

"Fitness" for feasible chromosomes is evaluated by calculating the total net value score (7) of the chromosomes representing release plans x. Each feature is associated with exactly one TNV function, which will give different values on different release dates.

For a given time horizon T and considering next $K = 3$ releases, for illustrative purposes, we assume a chromosome (release plan) refers to release dates RD = (140, 310, 440). If a feature's value function indicates values of {1,500, 1,370, and 1,200} for those dates for releases 1, 2, and 3, respectively, and the GA places the feature in release 2, then the feature value becomes 1,370. Fitness of the chromosome is then calculated by summing the value of all items (genes) in the release plan (chromosome).

The GA utilizes Or-Trees to create new generations. The Or-Tree approach is best documented in [24] and proves to be very efficient for finding solutions for highly constrained problems, which is the case with the release planning problem. During mating in the Or-Tree approach, alleles for children genes are chosen on a gene-by-gene basis from two parents, with any given gene position in a child containing the corresponding allele value from either (parent 1 OR parent 2) parent. Therefore, cross-over is completely random. The process is illustrated in Fig. 5.

Partial solutions are generated by adding genes and alleles to originally empty children, until a partial solution proves to be infeasible, at which time the Or-Tree is backtracked to try an allele from the other parent (see [24]). This process continues until a full feasible solution is obtained, or the tree fails.

Tree failure rates (failure to find a viable solution) were not explicitly measured and varied from run to run. In reality, in order to make the algorithm more efficient, full trees were never explored. We note that a full exploration of a tree representing a release plan with N features could involve 2^N nodes in the worst case, which soon becomes intractable. Instead, only a certain (specified by an input parameter to the GA) number of nodes were visited, and if a feasible solution was not discovered by the pre-determined node limit, the tree was discarded and a new one was generated and explored.

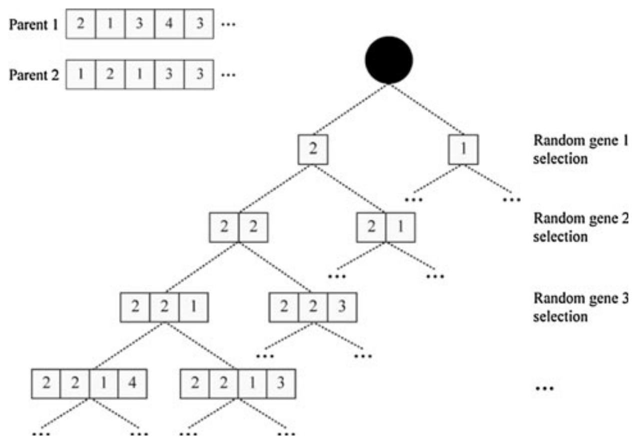

Fig. 5 Or-Tree structure

The observation here is that trees with a "bad" set of early gene alleles could result in very expensive, even full tree searches, and were best terminated early and tried again. There is a dynamic tension between the time of potentially "bad tree" searches versus the time of starting over again with a possibly better tree. Our implementation allows the number of nodes searched to be varied, and a switch can be turned onto monitor the general failure rate of the tree searches in order to adjust the variable controlling the nodes searched before a tree was abandoned. The variable is then "tuned" to try to optimize the speed of the algorithm in terms of completing successful tree searches.

In order to mitigate the local maxima effect, the GA was benchmarked against a specialized integer programming algorithms as implemented in the ReleasePlanner^{TM} decision support tool [28]. ReleasePlanner^{TM} provides plans of a proven degree of optimality (using upper bounds from solving relaxed problems). It also represents an implementation of the baseline model described in Sect. 3.1. The GA was calibrated until it was producing solutions that were closest to the quality of the integer program. Random genes (mutations) were introduced at a rate (10%) that would mitigate the tendency to focus on local maxima.

As a result of the calibration process, the GA input parameters included the following, listed along with the case study values for the parameters

- Random chromosome generation technique: Or-Tree
- Number of original parents: 50
- Number of mates per parent: 5
- Number of surviving offspring per generation: 45
- Number of fully mutant offspring per generation: 5
- Number of nodes visited in the Or-Tree before a search was abandoned and retried: 200
- Maximum number of generations: 300
- Number of generations to determine convergence: 10

The GA parameters described earlier were chosen via the benchmarking process described previously. The intention was to choose a set of GA parameters that not only produced good results, but also executed in a reasonable amount of time. For several input parameters, points were reached where increasing parameter values resulted in little increase in accuracy and the benefit/cost ratio deteriorated rapidly.

4.4 Phase 3: consolidation

Consolidation refers to the process of evaluating the qualified solution alternatives and making suggestions for refinement of the underlying models. An improved understanding of the problem typically results in modifying the parts of the underlying model. Often, these decisions reduce the size and complexity of the problem for the next iteration.

The result of the exploration phase is a set of trade-off solutions with varying release dates. Facing a small set of non-dominated solutions, the product manager and the associated stakeholders can get further insight into the problem structure and dependencies and move to the next iteration with a refined (better) model. Refinements can be related to any of the parameters, constraints, and objective formulated in DVB-RP in an attempt to qualify the model to better match reality.

The product manager has to either choose one of the solutions or to initiate another iteration with modified model parameters. Among the set of non-dominated trade-off solutions, the product manager can consider two key aspects to select one most preferred solution: (a) the trade-off values of the different solutions and (b) satisfaction of implicit concerns.

Implicit concerns are relevant aspects for making product release decisions, which are very hard or even impossible to be formulated by explicit constraints. As an example for an implicit concern, we consider the cohesiveness of the set of features offered for a specific release. While is it impossible to model and evaluate the cohesiveness of 2^N possible combinations of N given features for one release, the human experts might be able to tacitly judge the degree of cohesiveness of a given set of features of a specific release.

At this point, the release manager may choose a solution because it fulfills important implicit concerns sufficiently well. Alternatively, some of the input parameters to the problem can be adapted because of a better problem understanding obtained during the iteration. Changes can be made related to changing resource capacity or consumption, to modifying value functions, to varying the frequency of the release date sampling, or changes related to modifying risk measures. Once these input parameters are changed, the three iterations of problem solving as suggested by EVOLVE$^+$(DVB-RP) are repeated.

5 Case study

5.1 Case study setup

A case study was set up employing a benchmark release planning project at a large telecommunications company. The data used in this case study are synthetic data, derived from a benchmarking release planning project, modified to accommodate time-dependent feature values. The case study is intended to demonstrate the principal applicability of the proposed method. This is done by studying six planning scenarios and analyzing the process and the solutions as provided by EVOLVE$^+$(DVB-RP).

The planning includes 25 candidate features to be decided upon for their release placement. Each feature is actually a kind of mini project undergoing the key steps of a development cycle. In order to demonstrate the benefit gained from the consideration of varying release times and resource capacities, the original problem introduced in [29] has been modified and extended, using a time-dependent value function for each feature. In the same way, time-dependent resource capacity functions for each resource used for the variation of the release date were introduced. In this study, the resource capacity functions are modeled as linear functions.

Seven resources were defined. Six of the seven were variable human resources, and their capacities are stated in terms of units (person days) available per day. In addition to that, one non-time-dependent resource type (cost) was defined. Resource definitions are provided in "Appendix 1".

Features were divided into the two basic types discussed in Sect. 3.2—those for which daily income estimates could be made (market-driven features) and those for which daily income estimates were inappropriate, such as contract-driven features. Market-driven features Figs. 1, 3 provide examples of these graphs, with not all of the value curves for features being shown. "Appendix 2" provides the details of the features, resource consumption, and feature type (market or contract).

In addition, one coupling constraint and one precedence constraint were defined between features:

- China Feature 1 must precede China Feature 2
- USEast Feature 1 is coupled to USEast Feature 2

For the case study, three release dates intervals were defined such that:

$$\begin{aligned}
\mathrm{rd}(1) &\in [\mathrm{rd1}(1),\ \mathrm{rd2}(1)] = [130, 170]\\
\mathrm{rd}(2) &\in [\mathrm{rd1}(2),\ \mathrm{rd2}(2)] = [280, 320]\\
\mathrm{rd}(3) &\in [\mathrm{rd1}(3),\ \mathrm{rd2}(3)] = [430, 470]
\end{aligned}$$

The overall planning horizon considered is defined by $T = 2{,}445$ days, or roughly 6.6 years. In this case study, the planning horizon encompasses the time period in which daily gross income is estimated for market-driven features. However, income estimates past 1,220 days (about 3.3 years) are considered rough at best and can be replaced with single estimates (per feature) of the estimated residual income of a feature past the 3.3 year mark.

Consideration of risk is of key relevance in the problem formulation DVB-RP. Initially, the risk function (11) is defined as a linear additive function based on the difference between the actual release date rd(k) and latest release date rd2(k) agreed upon for all the releases $k = 1 \ldots K$. This is then normalized by dividing by the sum of the latest release dates rd2(k) minus the earliest possible release dates rd1(k):

$$\mathrm{Risk}(x, \mathrm{RD}) = \sum_{k=1 \ldots K} [\mathrm{rd2}(k) - \mathrm{rd}(k)] \Bigg/ \left\{ \sum_{k=1 \ldots K} [\mathrm{rd2}(k) - \mathrm{rd1}(k)] \right\} \quad (11)$$

One of the subsequent topics of investigation studied in Sect. 5.3 is the impact of varying risk functions on the results of planning.

5.2 The genetic algorithm and its implementation

For the case study, each generation of chromosomes was created from 50 parents, which generated 250 feasible offsprings. Each parent was mated with five other parents, and if mating failed to produce a feasible offspring, a new parent combination was tried, until mating was successful. The 45 chromosomes with the highest fitness scores from the pool of parents and children were passed as parents to the next generation. This proved to be far faster at converging on near optimal solutions than roulette selection.

In addition, each new generation contained 5 completely random (but feasible) solutions added to the 45 top solutions to provide a mutation factor in the mating of the next generation, which helped prevent convergence on local maxima. For each solution, a maximum of 300 generations could be created. However, if a solution remained unchanged after 10 generations, convergence was considered to have been obtained, and the process was terminated.

5.3 Questions

The case study intends to create more transparency into the results obtainable by the method. Without attempting to obtain external validity, six questions are studied. For each question, it is shown how the method is able to find answers. These answers are considered as contributions to decision support in the sense that they can help human decision makers arrive at their final decision [26]. The questions studied are:

Question 1: What impact does varying the release dates have on the time-dependent value of the generated release plans?
Question 2: How do time-dependent values compare with time-independent values?
Question 3: What does the trade-off between time-dependent value and inherent risk look like?
Question 4: Do earlier release dates always produce higher release plan values?
Question 5: What is the impact of varying time horizons for planning?
Question 6: How sensitive are the results related to changing risk parameters?

5.4 Findings for question 1

For answering question 1, the continuous release date parameter was discretized with a step-size of 10 days, which was used for the purpose of keeping the computational effort reasonable. All the combinations of possible release dates were analyzed. In total, this led to $5^3 = 125$ combinations of possible release dates. For the purpose of this investigation, this was considered sufficiently detailed to study the impact of the time-dependent functions in the presence of varying release dates.

When varying the release dates, the resource capacity functions have been varied correspondingly. For each of the release parameter settings, the GA was applied 10 times to mitigate the impact of randomness.

The best value of these runs was taken. There was a fairly high degree of concurrence between the 10 separate GA runs using different random numbers. The average release plan value difference between GA runs for specific release dates was 2.24%, with a standard deviation of 0.89%.

Figure 6 shows the results of the computations. The release dates are varied with the earliest release date changing the most rapidly and the latest release date changing most slowly. The unlabeled tic marks on the x-axis hide the variance of the earliest release date. There is a 5.1% difference in terms of release plan value between the highest valued release plan (Value(x, RD, T) = 10,272,830, for release dates RD = (130, 280, 430)), and the lowest valued release plan (Value(x, RD, T) = 9,747,785 for release dates RD = (170, 310, 430)). The average release plan value is 9,934,289.

In all, there are 25 segments where the first release date varies from 130 days to 170 days, while the second and third release dates remain constant. These segments are delineated by the labels on the x-axis of Fig. 6. Analysis of these 25 segments shows a definite trend. Of the 25 segments, the best release plan values (shown as peaks in Fig. 6) occur when the first release date is as early as possible (day 130), in 22 of the 25 segments. Similarly, the worst release plan values (shown as troughs) occur when the first release date is as late as possible (day 170) in 18 of the 25 segments. In the remaining 7 worst cases, the first release date is day 160.

We also analyzed the top 25 (top 20%) release plans in terms of the question at which of the five possible release dates (called 1…5 referring to the earliest to latest release

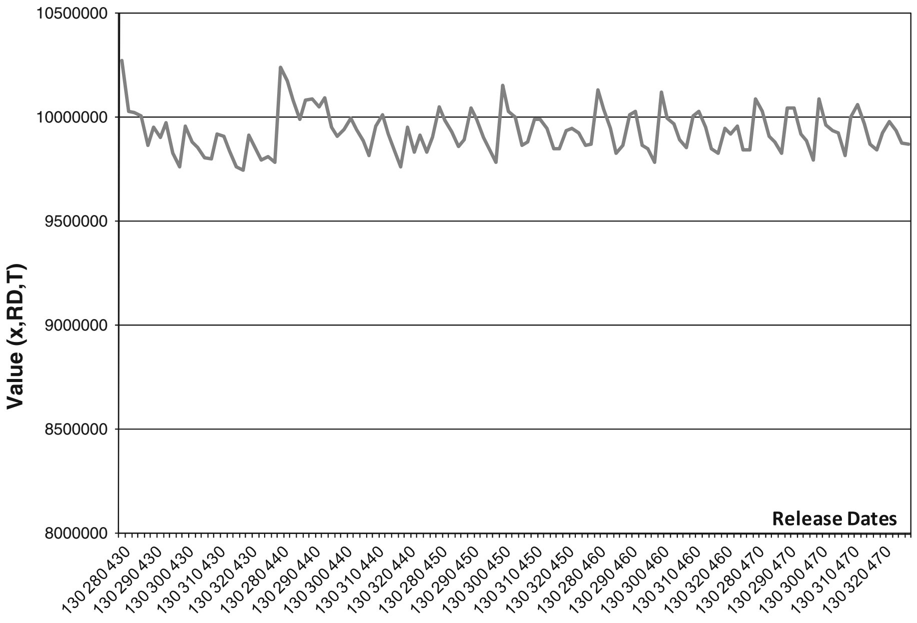

Fig. 6 Total net value variation in dependence of variation of release dates

date, respectively) the value was achieved. Figure 7 shows the results. There is a clear trend for the higher valued release plans to have early first and second release dates, while the third release date does not seem to have that much impact.

The conclusions of the analysis are not claimed to be externally valid. Instead, the purpose is to show, in general, the type of analysis that can be undertaken on any release planning dataset using time-dependent feature values, and possible insights that can be gained, and trends observed, from that analysis. This emphasizes the inherent decision support nature of this problem, where general insights and trends are often more important than specific solutions. In the case of the sample project, the analysis indicates that:

- up to a 5.1% improvement in release plan value can be obtained by utilizing flexible release dates in conjunction with time-dependent feature values,
- the release plans with the highest values tend to have the earliest first and second release dates. The third release dates do not seem to be as important.

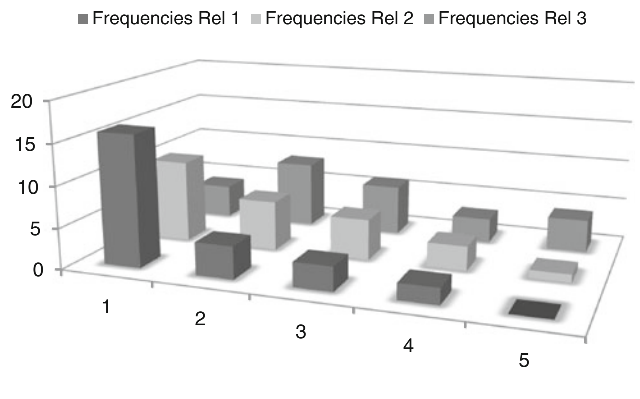

Fig. 7 Frequencies of occurrence for the five release date options for releases 1 to 3 (taken from the top 25 plans and time-dependent value function)

The above analysis fails to address two important issues for the sample dataset. First, given that the highest valued release plans are those with the earliest release dates in the defined release date interval, would even higher valued plans be obtained by making the first release date even earlier? This issue is addressed by the analysis of question 4 in this paper.

The second issue deals with the risk of releasing early. This type of risk is related to quality. Generally speaking, the earlier a set of features is released, the higher the chance of poor quality in those features. This issue is addressed by the analysis of various risk functions as specified in question 6 of this paper.

5.5 Findings for question 2

In question 2, we compare planning results between time-dependent values with time-independent values. In EVOLVE$^+$, features (n) are assigned time-independent scores as a results of a feature prioritization process performed by stakeholders. There may be several prioritization criteria such as overall business value, urgency (time to market), and impact if a feature is not present in a release. For all criteria, higher prioritization numbers (e.g., 9) indicate higher value. Once prioritized by stakeholders, these criteria can be combined using different formulas, from simple additive formulas to more complex formulas involving products, logs, weighting factors, etc. This overall set of formulas can be summarized as resulting in a number Prio(n, p) denoting the overall priority of feature $f(n)$ from the perspective of stakeholder p.

Stakeholders are also assigned weights $\lambda(p)$ in EVOLVE$^+$ on a similar nine-point scale depending on their overall importance, with 1 being the lowest importance and 9 the highest: $\lambda(p) \in \{1, 2, \ldots 9\}$. Given this, a weighted average priority for each feature n can be defined as:

$$\mathrm{WAP}(n) = \sum_{p=1\ldots q} \lambda(p) \cdot \mathrm{Prio}(n,p) \quad (12)$$

In addition, each release k, where $k \in \{1\ldots K\}$ has a weighting factor $\xi(k)$ for every feature placed in that release, with $\xi(k) \in \{0\ldots 9\}$. Typically, earlier releases have more importance than later releases, therefore, $\xi(1) > \xi(2) > \cdots > \xi(K)$. Release weights represent, in reality, a course-grained mechanism for simulating time-dependent value functions with variation just between the releases and having the same variation factors applied to all features.

Given the above, the objective function F in EVOLVE* can be formulated as:

$$\text{Maximize } F(x) = \sum_{k=1\cdots K} \xi(k)[\sum_{n:x(n)=k} \mathrm{WAP}(n)] \quad (13)$$

In EVOLVE$^+$(VAB-TRDB), the ordinal feature values described earlier for EVOLVE$^+$ are replaced with ratio values reflecting expected total net financial value of features. EVOLVE$^+$(VAB-TRDB) does not yet explicitly specify how these financial values are arrived at, except to say that they are reached by common consensus among qualified financial experts.

To conduct the intended comparison, all time-dependent feature value functions are fixed in EVOLVE$^+$ (DVB-RP) to be those values present at the middle of the first release interval. For the first release, this is at release date rd(1) = 150.

In order to compare with EVOLVE*, release weights are applied to the time-independent feature values. Instead of having release weights $\xi(k) \in \{0..9\}$, the weights are normalized such that $\xi(1) = 1$ and $\xi(2) \ldots \xi(K-1)$ are some fraction of 1. In this scenario, the release "weights" become penalties for later releases. This retains financial fidelity.

In the sample project, release weights were determined from averaging all feature value functions and using a linear approximation of this function applying least square technique. This results in average daily value deterioration for all features of about 0.143%.

In order to perform the comparative analysis, we define release factors. $\xi(1)$ is normalized to 1. Release 2 is assumed to occur 150 days later at day 300, and release 3 is assumed to occur at day 450. Given this, $\xi(2)$ is then set at $1 - (300 - 150)*0.143 = 0.785$, and $\xi(3) = 1 - (450 - 150)*0.143 = 0.57$. Summarizing, the three release weights are: $\xi(1) = 1$, $\xi(2) = 0.785$, $\xi(3) = 0.57$.

In this comparison, feature values are initially given as time independent and are affected only by the release weights as described earlier. However, resource capacities are still allowed to vary with time, as described in (5).

Similar to Fig. 6, the release dates are varied in Fig. 8 with the earliest release date changing the most rapidly and the latest release date changing most slowly. The unlabeled tic marks on the x-axis hide the variance of the earliest release date. There are 25 segments in Fig. 8 where the first release date varies from 130 to 170 days and the second and third release dates remain constant. Analysis of these 25 segments shows that of the 25 segments, the best release plan values (shown as peaks in Fig. 8) occur when the first release date is as late as possible (day 170), in 22 of the 25 segments. Similarly, the worst release plan values occur when the first release date is as early as possible (day 130) in 21 of the 25 segments.

However, as with the analysis in the previous section, when looking at the entire set of 125 release plans, it is possible for release plans with earlier first release dates to outperform release plans with later first release dates, when the second and third release dates are taken into consideration. In order to get a clearer picture, the top 25 (top 20%) release plans were evaluated. The results are shown in Fig. 9.

What is interesting to see is the contrast of this analysis and that done with time-dependent feature values. The tendency with time-dependent feature values (and expandable resource capacities) was that release plans with earlier releases had higher values. Conversely, when feature values are held constant, subjected only to release weights (penalties), and resource capacities are allowed to expand, the strong tendency was that release plans with later releases had the higher values.

This indicates that the effect of decreasing feature values over time outweighs the effect of increasing resource capacities over time in the sample project. This also infers three other conclusions:

1. For projects in which the sum of feature total net value decline over time is steep, earlier releases would be strongly favoured. However, other concerns, such as the risk of poor quality, might weigh against very early releases.
2. For projects in which the sum of feature total net value decline over time is mild, later releases might be favoured. But, again, other considerations, such as the risk that a competitor might beat one to the market, need to be accounted for.

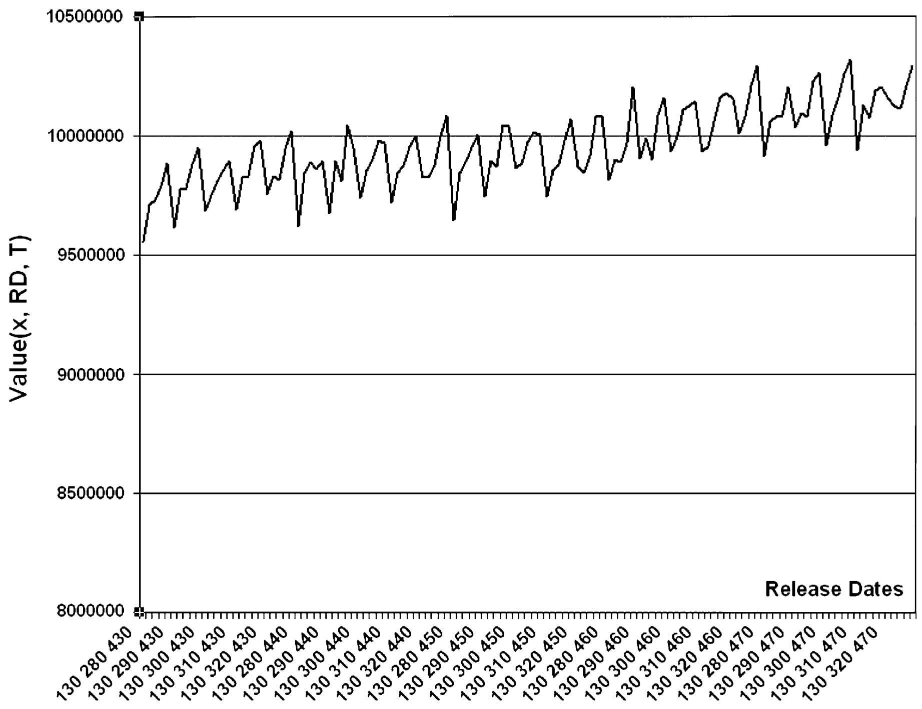

Fig. 8 Total net value variation in dependence of variation of release dates (fixed value function)

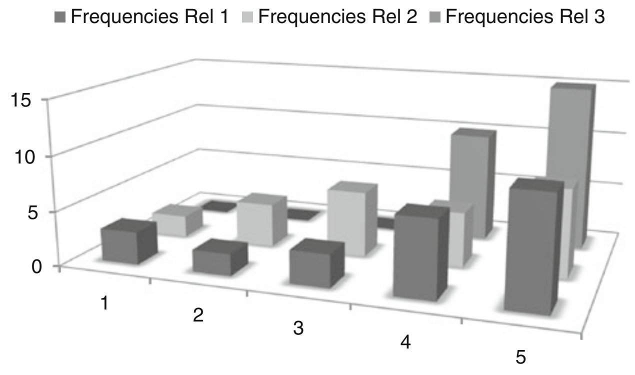

Fig. 9 Frequencies of occurrence for the five release date options for releases 1 to 3 (taken from the top 25 plans for fixed value function)

Table 1 Non-dominated solutions

Sol. #	Rd(1)	Rd(2)	Rd(3)	TNV	Risk
1	170	320	470	9872030	0.00
2	160	320	470	9876221	0.08
3	150	320	470	9935630	0.17
4	140	320	470	9982096	0.25
5	140	310	470	10060617	0.33
6	130	300	470	10089291	0.50
7	130	300	460	10121819	0.58
8	130	300	450	10154141	0.67
9	140	280	440	10179087	0.83
10	130	280	440	10244350	0.92
11	130	280	430	10272830	1.00

3. Some projects may have a degree of equilibrium between declining feature value, and increasing release value due to greater resource capacity. In these cases, other, possibly tacit concerns will dictate whether earlier or later release dates are favoured.

5.6 Findings for question 3

With this question, we investigate "what does the trade-off between time-dependent value and inherent risk look like?" The risk calculation is run in order to determine the non-dominated Pareto solutions for risk versus release plan value. The risk function employed is that specified in (11). For each pair (x, RD), the estimated risk of the release plan strategy w as determined. The results of these computations are shown in 10. We find the different layers of all the value-risk pairs generated for the different scenarios. Besides that, we observe that there is a set of 11 non-dominated solutions. All of these solutions have the defining property that there is no improvement possible for one criterion without deteriorating with respect to the other. Table 1 shows the exact coordinates of all 11 non-dominated trade-off solutions from Fig. 10. The selection of the final plan is expected to come from the set of non-dominated solutions. For that, some further investigation is performed in the consolidation phase of EVOLVE$^+$ (DVB-RP).

The trade-off curve is valuable information for the decision-maker. For the different levels of normalized risk, the highest achievable value (in conjunction with the plan to achieve this value) can be found. The highest achievable

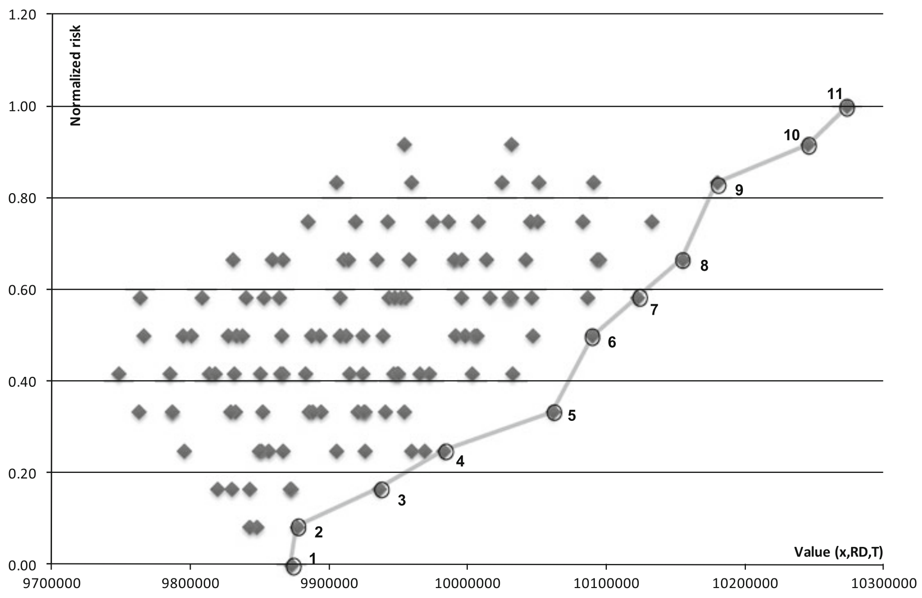

Fig. 10 Risk-value trade-off solutions

value is coupled with the highest risk. Vice versa, the lowest risk results in the lowest financial value.

5.7 Findings for question 4

In general, given "present value" issues, market competiveness, etc., the earlier the features selected for a release are offered, the higher their financial value tends to be. However, reducing the release date also means reducing the total resource capacity available to implement the features. This is another trade-off relationship which adds to the one related to risk. Question 4 has addressed the question if there is a break-even point in the value creation.

The reason for this enquiry was due to the fact that seven of the eight highest valued release plans shown in Fig. 11 have first release dates of 130 days, and in general, "130 days first release date solutions" dominate the solution set in terms of highest value. (16 of the top 25 solutions have first release dates of 130 days). As first release dates before 130 days were not sampled, this led to the question of whether reducing the first release date further for the sample project would continue to increase the release plan values, and whether there is a breakeven point for the first release date where release plan values peak, then drop-off again.

In order to resolve this issue, as stated previously, the first date interval was expanded to rd(1) ∈ [rd1(1), rd2(1)] = [110, 170]. The results of this investigation are shown in Fig. 11. Figure 11 shows the interval rd(1) ∈ [rd1(1), rd2(1)] = [110, 130], with the remaining rd(1) range [140…170] suppressed for clarity. The intervals rd(2) and rd(3) remain as before. What was discovered is that:

- Release plans with first release dates of 130 days have higher release plan values than release plans with first release dates of 110 days, in all cases tested in the sample project.
- Release plans with first release dates of 130 days have higher release plan values than release plans with first release dates of 120 days in 21 of the 25 cases in the sample project.
- In the four cases where 120 days first date solution values exceeded 130 days first date solution values, the value increases were insignificant—0.2, 0.09, 0.02, and 0.6%.

The seven highest valued release plans continued to be those with first release dates of 130 days.

For the sample project,, it was determined that in most cases, having a first release date of 130 days for this project represented the break-even point where release plan values tended to be highest, and sampling earlier dates was not productive.

5.8 Findings for question 5

In question 5, we investigate the impact of variation of the time horizon on planning results. The sample project has a time horizon of 2,445 days, or about 6 2/3 years, for market-driven features. In this case, the "time horizon" means the period of time in which the daily gross income for the market-driven features is estimated. Income estimates past the half-way mark of this horizon (past 1,220 days or roughly 3 1/3 years) are made for quarterly periods rather than monthly periods due to the greater uncertainty of these estimates.

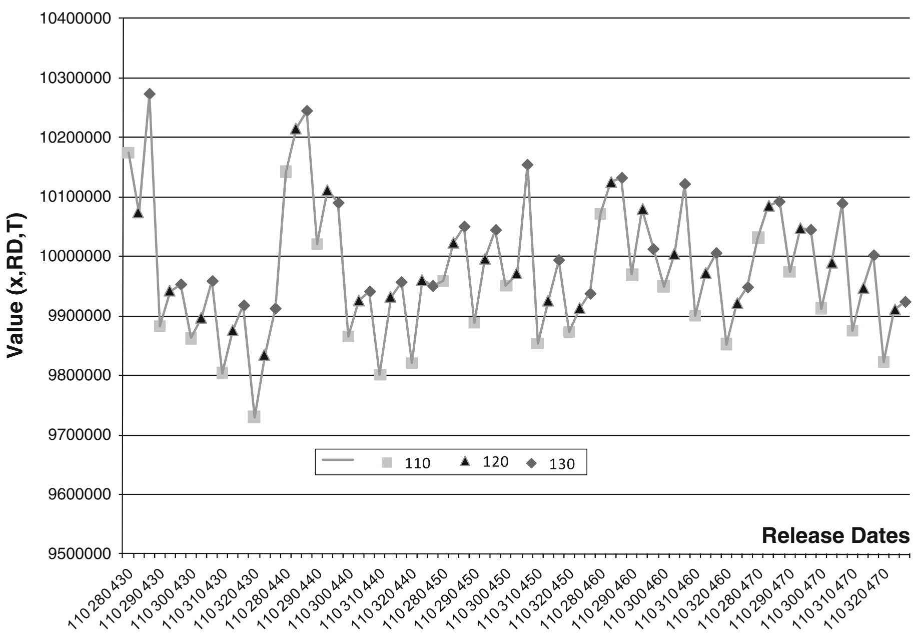

Fig. 11 Total net value in dependence of rd(1)

Due to this inherent uncertainty, the financial estimates past 3 1/3 years would be considered rough at best and would probably represent residual income from the features past the point where they are not expected to be major sources of income. Exceptions to this might be features that are expected to gain acceptance over time, e.g., feature f(5) in Fig. 1.

Contract-driven features are not affected by time horizons because the value of a contract-driven feature is fixed on any given day and is not affected by future predicted income. This is a somewhat simplified model used in this paper and ignores factors such as lost future business due to dissatisfied customers.

The sample project contains 19 market-driven features and 6 contract-driven features. The 19 market-driven features all have their estimated gross daily income values dropped to zero, or very near zero, by day 2,445 as shown in Fig. 12 below.

It can be seen that the value functions from day 1,220 to 2,445 for these features are similar (in form) but not identical. For the purpose of empirical evaluation, the time horizon of the sample project was truncated at days 1,460 and 1,830. Day 1,830 was chosen because it lies about halfway between 1,220 and 2,445, i.e., ¾ the way through the existing time horizon. It is also very close to 5 years. Day 1,460 was chosen because it is exactly 4 years.

The intention here is to compare the plots representing the release plan values for the different release dates against one another. The assumption is that identical plots would indicate identical sets of release plans, differing only by a constant value factor, represented by the vertical position of the plots on the graph.

In order to better compare the release plan values, a constant number will be added to each truncated release plan value in order to facilitate the comparison.

Figure 13 shows the comparison of the adjusted 1,830 days release plan values (dashed line) with the 2,445 days release plan values (solid line). In both sets of release plans, the release date combination with the highest release plan value is (130, 280, 430).

The constant factor to be added to the 1,830 days release plans was determined by calculating the integer that, when added to each 1,830 days release plan value, came closest to creating an average deviation from the 2,445 days plan value of zero.

As can be seen in Fig. 13, the alignment of the two sets of release plan value plots is close. An assessment of the release plan values shows that of the 125 value points pairs (for the 1,830 and 2,445 plans), 120 lie within 1.1% of each other. Of the remaining 5 points, all lie within 1.5% of each other.

In terms of the release plans themselves, the first 25 release plans from both sets were analyzed in terms of their structural similarity. The measures were employed—Euclidean and Hamming distances. The Euclidean distance helps one obtain an understanding of how different two release plans are in terms of the placement of features in releases. The Hamming distance simply shows how many features have different release placements in two release plans without saying how different the release placements are.

Surprisingly, only one release date vector RD = (130, 310, 430) had identical release plans for both time horizons. Among all the 25 pairs of plans, the Euclidean distance varied from 0 (for just a single case) to 5.57.

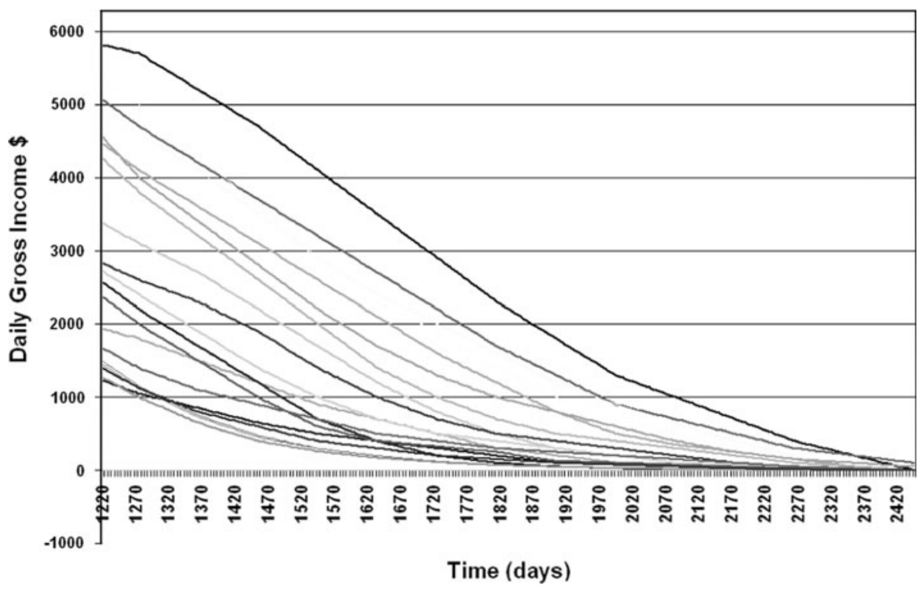

Fig. 12 Daily gross income functions past day 1,220

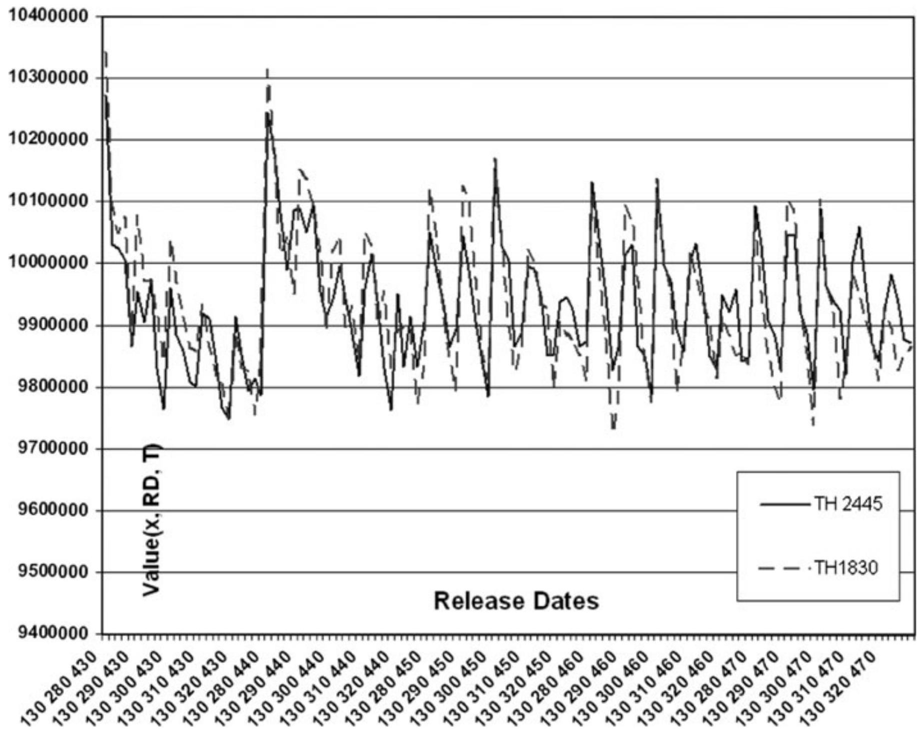

Fig. 13 Adjusted 1,830 days release plan values compared to 2,445 days values

Similarly, the Hamming distance varied from 0 (for just a single case) to 14.

The interesting results from this comparison is that while the release plan date pattern is very similar between the two time horizons, as seen in Fig. 13, the release plan pairs vary quite a bit, at least for the 25 sampled pairs of plans.

A similar comparison was done on release plans with a time horizon of 1,460 days (4 years) and release plans with the 2,445 days time horizon. As with the previous example, a constant factor was added to the 1,460 days release plan values that came closest to creating an average deviation from the 2,445 days plan value of zero. The resulting plot is shown in Fig. 14.

It can be seen from Fig. 14 that the 1,460 days values are not nearly as good a fit with the 2,445 days values as were the 1,830 days values. This is somewhat predictable, as the 1,830 days release plan set truncated only 25% of the 2,445 days time horizon, while the 1,460 days release plan set truncates 40% of the time horizon.

Again, 25 release plans from both sets were analyzed in terms of their structural similarity (In this case, release

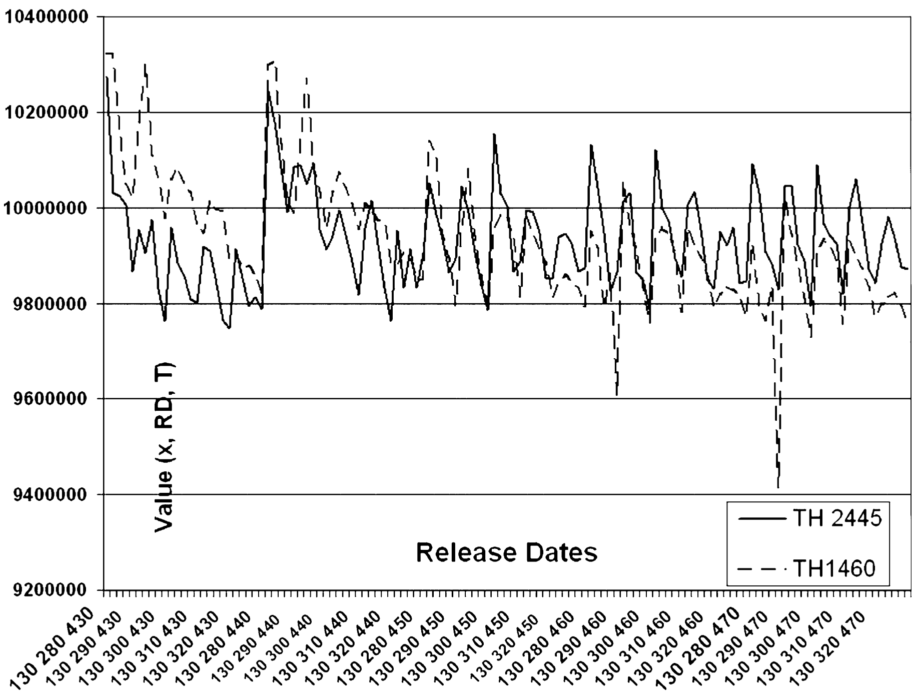

Fig. 14 Adjusted 1,460 days release plan values compared to 2,445 days values

plans 51–74 were analyzed, because that is where the curves appear to be the closest). There were no pairs of release plans in the 25 pairs analyzed that were identical. The comparison of the Euclidian distance, Hamming distance, and relative distance yielded the following results for the 25 compared pairs of release plans:

- The Euclidean distance between pairs of plans varied from 4.00 to 7.68.
- The Hamming distance between pairs of plans varied from 6 to 15.

The significant findings of these two investigations, as they to the sample project, are the following:

1. Contracting the time horizon of the project by truncating that horizon retained the basic pattern of release values across release dates found with the original time horizon, although that pattern seemed to deteriorate the more the time horizon was truncated.
2. While the basic release value pattern remained similar between the full time-line project and truncated time-line projects, the release plans themselves did not. For the 20% of release plans sampled, there were significant differences between plans, as measured by the Euclidean and Hamming distances.

5.9 Findings for question 6

The final topic of investigation is related to the question of how sensitive the results are related to changing risk parameters. For that purpose, three additional risk functions were defined and compared with the baseline one.

The first additional risk function is similar to (11) but weighs $\omega(k)$ in the earlier releases to indicate higher risk in releasing too soon in those earlier releases. It too is normalized in a manner similar to (11) using rd1(k) and rd2(k) for the release k start and end dates, respectively:

$$\mathrm{Risk}(x, \mathrm{RD}) = \left\{ \sum_{k=1\ldots K} (\omega(k)[\mathrm{rd2}(k) - \mathrm{rd}(k)] \right\} \Bigg/ \left\{ \sum_{k=1\ldots K} \omega(k)[\mathrm{rd2}(k) - \mathrm{rd1}(k)] \right\} \tag{14}$$

In the case study, we are using $\omega(1) = 3$, $\omega(2) = 2$, and $\omega(3) = 1$. This corresponds to the assumption that the risk in earlier releases is considered to be higher than the one for later releases.

The second additional risk function is quadratic in nature and takes the following normalized form.

$$\mathrm{Risk}(x, \mathrm{RD}) = \sum_{k=1\ldots K} \omega(k)[\mathrm{rd2}(k) - \mathrm{rd}(k)]^2 \Bigg/ \left\{ \sum_{\mathrm{k}=1\ldots\mathrm{K}} \omega(k)[\mathrm{rd2}(k) - \mathrm{rd1}(k)]^2 \right\} \tag{15}$$

In the case study, we are using $\omega(1) = \omega(2) = \omega(3) = 1$ for simplicity reasons.

The third additional function is exponential in nature. It does not take the generalized form expressed in (6) but was added in for comparison with the other functions.

The exponential function takes the following normalized form:

$$\text{Risk}(x, \text{RD}) = \sum_{k=1\ldots K} \exp[(\text{rd1}(k) - \text{rd}(k))/\omega(k)]/K \quad (16)$$

For the case study, we have used $\omega(1) = \omega(2) = \omega(3) = 10$ to allow best comparison between the functions.

These risk functions are compared in Fig. 15 The items to focus on in the figure are the Pareto solutions for the four different risk functions. For a given risk function, these solutions have the defining property that there is no improvement possible for one criterion without deteriorating the other criterion.

In Fig. 15, the plots for the four risk functions are shown simultaneously to facilitate detection of commonalities among the Pareto solutions, and commonalities do emerge. Table 2 shows the 17 non-dominated solutions showing up in the four Pareto fronts. Solutions 1, 4, 13, 14, 15, 16, and 17 show up in all four Pareto fronts for all four risk functions. Only, four of the solutions are unique to a particular risk function: solutions 3, 5, 8, and 10.

In Table 2, rf(1) is the linear risk function (11), rf(2) is the weighted linear risk function (14), rf(3) is the quadratic risk function (15) and rf(4) refers to the exponential risk function (16). rd(1), rd(2) and rd(3) are release dates 1, 2, and 3, respectively. Interestingly, while there are a total of 47 risk/release plan value pairs (non-dominated solutions) in the four Pareto fronts, there are only 17 unique release plans solutions generating these 47 risk-value pairs for the four different risk functions.

Identical release plans in different Pareto fronts align vertically in Fig. 15 as the x-coordinate holds the release plan values. It can be seen, for example, that the five rightmost Pareto solutions for the four risk functions all involve the same release plans, as indicated by their vertical alignments. The similarities in the Pareto release plans across risk functions seems to indicate that there is some degree of stability in release plans in the Pareto solutions across different risk functions. Again, these results apply only to the sample project, and such stability cannot automatically be inferred to apply to other datasets.

Table 2 Comparison of non-dominated solutions for four types of risk functions

Sol #	rd(1)	rd(2)	rd(3)	TNV	rf(1)	rf(2)	rf(3)	rf(4)
1	170	320	470	9872030	0.00	0.00	0.00	0.02
2	160	320	470	9876221	0.08		0.02	0.03
3	170	320	440	9905030		0.13		
4	150	320	470	9935630	0.17	0.25	0.08	0.06
5	150	320	460	9959144		0.29		
6	150	310	470	9968574		0.33	0.10	0.07
7	140	320	470	9982096	0.25	0.38		
8	150	300	450	10004095				0.14
9	140	310	470	10060617	0.33		0.21	0.15
10	170	280	440	10085502		0.46		
11	130	300	470	10089291	0.50		0.42	
12	150	290	440	10094668		0.63		0.29
13	130	300	460	10121819	0.58	0.71	0.44	0.40
14	130	300	450	10154141	0.67	0.75	0.50	0.42
15	140	280	440	10179087	0.83	0.83	0.71	0.58
16	130	280	440	10244350	0.92	0.96	0.85	0.79
17	130	280	430	10272830	1.00	1.00	1.00	1.00

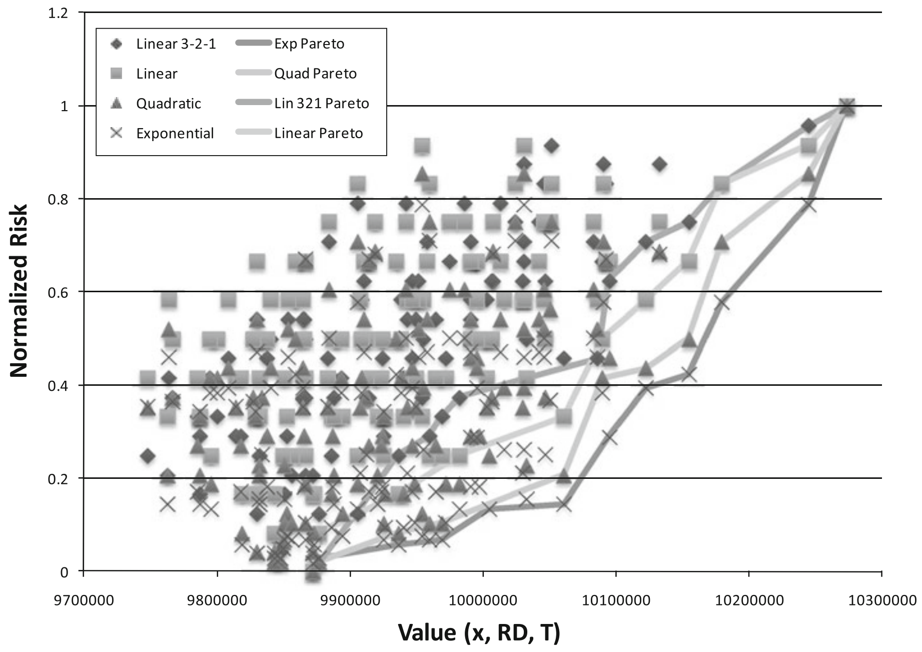

Fig. 15 Risk-value trade-off solutions

6 Discussion of results

6.1 Applicability of results

Release planning in general is of key importance to incremental product development [3, 14]. The idea of evolutionary problem solving integrating human and computational capabilities accepts the fact that software engineering decision problems are uncertain. We argue that the emphasis is not to find the ultimate one best solution, but to qualify the process to receive good solution alternatives able to address the real-world problem including some of its implicit concerns [25].

In general, planning is an ongoing process, which needs several iterations and re-planning steps to adjust to new parameters and regulations. In case of EVOLVE$^+$ (DVB-RP), more detailed information is used in comparison with the former EVOLVE$^+$ approach. This implies the chance of achieving more detailed results, but also implies potentially more effort to achieve these results.

When-to-release decisions are trade-off decisions by their very nature. The product manager needs to balance the possible gain (expressed by the value function) against the potential risks of delivering (too) early. This risk here is essentially related to quality and customer acceptance. Although difficult to express formally, our assumption is that the risk is higher the earlier the release date is chosen.

From applying specialized GAs, the user obtains a set of good (on average) solutions that allows one to look at them from the perspective of implicit concerns. This flexibility is further emphasized by providing trade-off solutions. However, this requires stronger involvement of the human experts as well. The presence of a set of non-dominated solutions at each of the iteration creates more flexibility for the product manager. However, it also requires additional analysis of all these candidate solutions from the perspective of the implicit concerns.

We are aware of the situation that the availability of the data is a critical pre-requisite and not automatically achieved by all organizations. However, we think this should not prevent us from developing a method which is using them. EVOLVE$^+$(DVB-RP) is not supposed to be a new silver bullet [8] but a valid alternative for mature organizations [9] more likely to provide the level of detail of information as requested. If this level of detail is not yet available, simplified release planning models having been applied successfully in practice [29] already can be taken as a first step.

There are a number of assumptions that need to be fulfilled to apply the given approach EVOLVE$^+$(DVB-RP) in real world. The granularity of features is assumed to allow the projection of estimated value functions. The more fine-grained the feature becomes, the more difficult it is to provide a reliable projection. Besides the feature value functions, feature effort estimates are needed, as well as and the (time-dependent) amount of available capacities. The feature value functions can be approximated by piece-wise linear functions resulting from monthly and quarterly estimates, which mirrors the real-world approach of soliciting stakeholder opinion about monthly and quarterly income potential.

The risk for early releases is hard to judge as well. Evaluation of risks can be performed as a result of the interaction of selected stakeholders. Even in case they are not 100% correct (they will never be), it allows one to get at least approximate solutions, which is often sufficient in software engineering.

6.2 Validity

It must be noted that the data used in this case study were synthetic data, derived from a benchmarking release planning project, modified to accommodate time-dependent feature values. Optimal release dates in other projects would be completely dependent on the data specific to those projects.

The results of the case study are more explorative than confirmative. One threat to internal validity stems from the genetic algorithm used to generate release plans in the research. Genetic algorithms are, by nature, non-deterministic, which can make a determination of their validity difficult.

Another threat (called construct validity) relates to the relationship between theory and observation. How to make sure the proposed model reflects real world sufficiently well? Although the model accommodates more advanced problem situations, it is still an approximation. This is done not only for computational reasons, but also for keeping the modeling effort reasonable, Addressing implicit concerns would potentially require to look at the order of $O(2^N)$ constraints formulated for all possible subsets of features. The same is true for dependencies between the value and/or resource consumption of features.

However, the overall approach is considered valid, as the approach is based on a proven baseline model which has demonstrated its applicability in industry in more than 20 industry projects. The extended model can accommodate any piece-wise linear value function. The question remaining is "how much of an impact would including value functions in release planning have in real-world projects, using real-world data and value functions, versus approaches that do not incorporate these data and value functions?" This paper describes mechanisms for exploring this issue but leaves the issue itself for future research.

7 Conclusions and future work

Software release planning is a problem with significant impact on the success or failure of software product development. A systematic approach is likely to generate plans achieving higher business value. In this paper, we have extended former efforts to formalize the problem and to apply both computational intelligence and expert opinion in a hybrid approach by allowing (a) time-dependent value functions for each feature, (b) flexible release dates varying within a given interval, and (c) adjusted time-dependent resource capacities.

We have developed a flexible and hybrid solution approach that essentially relies on solving a sequence of planning problems using a genetic algorithm. The case study demonstrated the principal applicability of this process. The six scenarios show the strengths of the method in providing support for making release decisions.

The intention of this paper is not to evaluate the GA for efficiency and/or quality of solutions based on different input parameters, which would lead to an evaluation of a combinatorial explosion of parameter combinations and their results. Rather, once calibration achieved satisfactory results, those parameters were set as the benchmark for all GA runs.

In future research, we want to compare the performance of the Or-Tree approach with other GA approaches. We also aim at allowing stronger interaction between stakeholders in the process of creating and evaluating the proposed solutions as well as for defining the key parameters of the problem.

Acknowledgments The authors would like to thank both the Natural Sciences and Engineering Research Council (NSERC) (Discovery Grant 250343-07) and the Alberta Informatics Circle of Research Excellence (iCORE) for their financial support of this research. In addition, further thanks goes to Jörg Denzinger of the University of Calgary for his help with designing the Or-Tree based genetic algorithm used in this paper. Last, but not least, the detailed comments provided by the reviewers of the paper have been of significant help.

Appendix 1

See Table 3.

Table 3 Resources and their capacities

Resource name	Type	Units	Release 1 capacity	Release 2 capacity	Release 3 capacity	Time-dependent category
BTS SW developers	Effort	Person days	1,800	900	1,400	Time-dependent
BTS HW developers	Effort	Person days	1,104	960	960	Time-dependent
BSC SW developers	Effort	Person days	2,160	1,680	1,680	Time-dependent
MTX SW developers	Effort	Person days	960	960	960	Time-dependent
Testers	Effort	Person days	1,680	1,680	1,680	Time-dependent
Documentation writers	Effort	Person days	600	480	600	Time-dependent
Capital req.	Cost	Dollars (1 K)	4,800	4,800	4,800	Constant

Appendix 2

See Table 4.

Table 4 Case study features and their characteristics

	Feature	Resource consumption							Feature type	Time-dependent value tendency
		BTS SW Dev	BTS HW Dev	BSC SW Dev	MTX SW Dev	Testers	Documentation	Cost ($ 1,000)		
1	Cost reduction of transceiver	150	200	120	0	200	60	1,000	Market	Daily—rise to day 885, then gradual but accelerating decline
2	16 sector, 12 carrier BTS for China	400	300	150	150	200	150	1,000	Contract	TNV—Sudden drop-off at day 280
3	Expand memory on BTS controller	75	120	10	0	75	20	200	Market	Daily—general deterioration
4	Next generation BTS 'in a shoebox'	450	350	375	125	500	200	150	Market	Daily—general deterioration
5	Pole Mnt packaging	400	180	300	50	400	150	500	Contract	TNV—sudden drop-off at day 300
6	FCC emissions regulatory change	400	120	100	0	200	10	200	Market	Daily—general deterioration, increasing in rate at day 365
7	India BTS variant	575	420	400	200	250	200	750	Market	Daily—general deterioration
8	Software quality initiative	450	0	100	50	400	5	0	Market	Daily—general deterioration
9	US East feature 1	100	0	400	100	40	100	0	Contract	TNV—Sudden drop-off at day 445
10	US East feature 2	200	0	400	150	50	50	25	Contract	TNV—Sudden drop-off at day 440
11	US East feature 3	400	0	100	100	40	20	100	Market	Daily—general deterioration
12	US East feature 4	150	0	400	125	400	150	1,000	Market	Daily—general deterioration
13	US East feature 5	75	180	225	225	300	60	750	Market	Daily—general deterioration
14	China feature 1	50	0	250	140	200	60	500	Market	Daily—general deterioration
15	China feature 2	60	10	120	120	190	40	200	Market	Daily—general deterioration
16	China feature 3	75	75	300	120	450	50	500	Market	Daily—rise to day 855, then gradual but accelerating decline
17	China feature 4	0	0	100	150	100	50	0	Market	Daily—rise to day 735, then gradual but accelerating decline
18	China feature 5	250	100	400	400	400	50	300	Market	Daily—rise to day 855, then gradual but accelerating decline
19	India market entry feature 1	200	100	250	250	250	100	500	Contract	TNV—sudden drop-off at day 140
20	India Market entry feature 2	0	0	300	250	250	100	300	Contract	TNV—sudden drop-off at day 140
21	India market entry feature 3	100	100	150	100	300	25	1,200	Market	Daily—general deterioration
22	Common feature 01	100	0	250	100	200	0	50	Market	Daily—holiday peaks
23	Common feature 02	0	0	100	250	150	50	0	Market	Daily—holiday peaks
24	Common feature 03	200	0	150	0	100	20	0	Market	Daily—holiday peaks
25	Common feature 04	100	0	300	200	200	30	50	Market	Daily—holiday peaks

References

1. Amandeep A, Ruhe G, Stanford M (2004) Intelligent support for software release planning. In: 5th international conference on product focused software process improvement, pp 248–262
2. Aurum A, Wohlin C (2003) The fundamental nature of requirement engineering activities as a decision-making process. Inf Softw Technol 45:945–954
3. Barney S, Aurum A, Wohlin C (2008) A product management challenge: creating software product value through requirements selection. J Syst Architect 54:576–593
4. Bhawnani P, Far B, Ruhe G (2005) Explorative study to provide decision support for software release decisions. In: IEEE international conference on software maintenance, ICSM'2005, pp 617–620
5. Bhawnani P, Ruhe G, Kudorfer F, Meyer L (2006) Intelligent decision support for road mapping—a technology transfer case study with siemens corporate technology. In: Workshop on technology transfer in software engineering, pp 35–40
6. Boehm B (1985) A spiral model of software development and enhancement. In: Proceedings of the International Workshop Software Process and Software Environments, ACM Press
7. Briand L, Wieczorek I (2002) Resource estimation in software engineering. In: Marciniak J (ed) Encyclopedia of software engineering, vol. 2. Wiley, pp 1160–1196
8. Brooks F (1987) No silver bullet: essence and accidents of software engineering. In: Proceedings of the IFIP, IEEE Computer Science Press, pp 1069–1076
9. Chrissis MB, Konrad M, Shrum S (2006) CMMI—guidelines for process integration and product improvement. Addison-Wesley, Reading
10. Clarke J, Dolado JJ, Harman M, Hierons R, Jones B, Lumkin M, Mitchell B, Mancoridis S, Rees K, Roper M, Shepperd M (2003) Reformulating software engineering as a search problem. IEE Proc Softw 150:161–175
11. Cohen MA, Eliashberg J, Ho T (1996) New product development: the performance and time-to-market tradeoff. Manage Sci 42:173–186
12. Cohn M (2006) Agile estimating and planning. Prentice Hall, Englewood Cliffs
13. Denne M, Cleland-Huang J (2004) The incremental funding method: data-driven software development. IEEE Softw 21:39–47
14. Ebert C (2007) The impacts of software product management. J Syst Softw 80:850–861
15. Gilb T (1989) Principles of software engineering management. Addison Wesley, Longman
16. Greer D (2004) Decision support for planning software evolution with risk management. In: 16th international conference on software engineering and knowledge engineering SEKE'04, Banff, pp 503–508
17. Harman M (2007) The current state and future of search based software engineering. In: International conference on software engineering ICSE 2007, series on the future of software engineering, Minneapolis, pp 342–357
18. Jung H-W (1998) Optimizing value and cost in requirements analysis. IEEE Softw 15:74–78
19. Karlsson J, Ryan K (1997) Prioritizing requirements using a cost-value approach. IEEE Softw 14:67–74
20. Larman C, Basili V (2003) Iterative and incremental development: a brief history. IEEE Comput, pp 47–56
21. Maurice S, Ruhe G, Saliu O, Ngo-The A, Brassard R (2005) Decision support for value-based software release planning. Value Based Softw Eng 253–268
22. McElroy J, Ruhe G (2007) Software release planning with time-dependent value functions and flexible release dates. In: 11th IASTED international conference on software engineering and applications 2007, Cambridge, pp 429–438
23. Momoh J, Ruhe G (2006) Release planning process improvement—an industrial case study. Softw Process Improv Pract 11:295–307
24. Nakano R, Yamada T (1992) A genetic algorithm applicable to large-scale job-shop problems. In: Parallel problem solving from nature 2, Elsevier, Amsterdam, pp 281–290
25. Ngo-The A, Ruhe G (2008) A systematic approach for solving the wicked problem of software release planning. Soft Comput 12:95–108
26. Ngo-The A, Ruhe G (2005) Decision support in requirements engineering. In: Aurum A, Wohlin C (eds) Engineering and management software requirements, Springer, Berlin, pp 267–286
27. Ngo-The A, Ruhe G (2009) Optimized resource allocation for software release planning. IEEE Trans Softw Eng 35:109–123
28. ReleasePlanner^{TM}, www.releaseplanner.com, Expert Decisions Inc., February 2010
29. Ruhe G, Ngo-The A (2004) Hybrid intelligence in software release planning. Int J Hybrid Intell Syst 1:99–110
30. Saaty TL (1980) The analytic hierarchy process. McGraw-Hill, New York
31. Saliu O, Ruhe G (2005) Supporting software release planning decisions for evolving systems. In: 29th IEEE/NASA software engineering workshop, Greenbelt, MD, USA
32. van den Akker M, Brinkkemper S, Diepen G, Versendaal J (2008) Software product release planning through optimization and what-if analysis. Inf Softw Technol 50:101–111
33. Wiegers K (2003) Software requirements. Microsoft Press, Redmond