# Software Release Planning

*Günther Ruhe*

University of Calgary

2500 University Drive NW

Calgary, AB T2N 1N4, Canada

++1 403 220-7692

ruhe@ucalgary.ca

## Abstract

Incremental software development replaces monolithic-type development by offering a series of releases with additive functionality. To create optimal value under existing project constraints, the question is what should be done when? Release planning is giving the answer. It determines proper priorities and assigns features to releases. Comprehensive stakeholder involvement ensures a high degree of applicability of the results. The formal procedure of release planning is able to consider different criteria (urgency, importance) and to bring them together in a balanced way. Release planning is based on (estimates of) the implementation effort. In addition, constraints related to risk, individual resources necessary to implement the proposed features, money, or technological dependencies can be easily adopted into the release planning approach presented in this article..

Releases are known to be new versions of an evolving product. However, the idea of a release is not restricted to this, but can be applied to any type of periodic development where a release would correspond to an annual or quarterly time period. The special case of one release called prioritization is of even larger applicability wherever competing items have been selected under additional constraints

An informal and later a formal problem description of the release planning problem is given. Ad hoc or just experience-based planning techniques are not able to accommodate size, complexity and the high degree of uncertainty of the problem. Plans generated in this way will typically result in unsatisfied customers, time and budget overruns, and a loss in market share. As a consequence of the analysis of the current state-of-the practice, we propose a more advanced approach based on the strengths of intelligent software engineering decision support.

Existing release planning methods and tool support are analyzed. An intelligent tool support called ReleasePlanner™ is presented. The web-based tool is based on an iterative and evolutionary solution procedure and combines the computational strength of specialized optimization algorithms with the flexibility of intelligent decision support. It helps to generate and evaluate candidate solutions. As a final result, a small number of most promising alternative release plans are offered to the actual decision-maker. Special emphasis is on facilitating what-if scenarios and on support of re-planning. Different usage scenarios and a case study project are presented. Practical experience from industrial application of ReleasePlanner™ is included as well. Future directions of research are discussed.

***Keywords:*** Software Engineering Decision Support, Incremental Software Development, Requirements Management, Releases, Feature Engineering, Stakeholder, Optimization Algorithms.

## 1. Introduction

Requirements management in general is concerned with the control of system requirements that are allocated to software to resolve issues before they are incorporated into the software project. It aims to accurately adjust plans and cost estimates as the requirements change, and to prioritize requirements according to their importance and their contribution to the final value of the product. There is very good reason to significantly improve the maturity of these processes. According to the Standish Group report [Standish Research Group 02], the three leading causes of quality and delivery problems in software projects are related to requirements management issues: Lack of adequate user input, incomplete requirements and specifications, and changing requirements specifications.

A software release is a collection of new and/or changed features that form a new product. Release planning for incremental software development assigns features to releases such that most important technical, resource, risk and budget constraints are met. Without good release planning 'critical' features are jammed into the release late in the cycle without removing features or adjusting dates. This might result in unsatisfied customers, time and budget overruns, and a loss in market share [Penny 02]. "Developing and releasing small increments of requirements, in order for customers to give feedback early, is a good way of finding out exactly what customers want, while assigning a low development effort" [Carlshamre 02].

In this article, we focus on features as main characteristics of a release plan. Features are considered to be "a logical unit of behaviour that is specified by a set of functional and quality requirements" [Gurp, Bosch & Svahnberg 01]. In other words, features are an abstraction from requirements that both customers and developers understand. Most of the topics discussed in this article are applicable to both the original requirements as well as to their aggregation into features. There is a growing recognition that features act as an important organizing concept within the problem domain and as a communication mechanism between users and developers [Turner et al. 99]. They provide an efficient way to manage the complexity and size of requirements.

The concept of a feature is applicable and important for any software development paradigm. However, it is especially important for any type of incremental product development. Features are the "selling units" provided to the customer. Incremental development has many advantages over the traditional waterfall approach. First, prioritization of features ensures that the most important features are delivered first. This implies that benefits of the new system are realized earlier. Consequently, less important features are left until later and so, if the time or budget is not sufficient, the least important features are the ones most likely to be omitted. Second, customers receive an early version of the system and so are more likely to support the system and to provide feedback on it. Third, the schedule and cost for each delivery stage are easier to estimate due to smaller system size. This facilitates project management and control. Fourth, user feedback can be obtained at each stage and plans can be adjusted accordingly. Fifth, an incremental approach is sensitive to changes or additions to features.

Agile methods [Cockburn 02] have capitalized on the above advantages. In Extreme Programming [Beck 01], a software product is first described in terms of 'user stories'. These are informal descriptions of user requirements. In the planning process called 'Planning games', these stories are prioritized using the perceived value to the user and assigned to releases. Based on estimates of how long each story in an increment will take to implement, an iteration plan is developed for delivering that release. Each increment (or release) is a completed product of use to the customer. At any time, new stories may be added and incorporated into future releases. The whole planning procedure is mainly based on communication and the underlying assumption that the main stakeholders are physically present at the meeting. It is further assumed that they are able and willing to achieve a compromise of their typically conflicting opinions. However, planning games are unlikely to generate transparent release plan alternatives for larger problems. They are unable to address all the resource, technological or risk constraints in an appropriate manner

This article is structured into eight sections. Following the introduction, an informal and later on a formal problem statement is presented in Section 2. A discussion of the relationship to other software disciplines and techniques is included in this part as well. Difficulties of the problem are presented to derive an appropriate understanding for its solution approach. Requirements prioritization and planning games techniques are presented in Section 4. Intelligent decision support for release planning as realized in the tool ReleasePlanner™ is presented in Section 5. As a consequence of the degree of uncertainty, the problem size, and the problem complexity, we propose to rely on the synergy between computational strength and the experience and intelligence of the human decision maker. This is the guiding principle of

the solution approach called EVOLVE* presented in this section. This includes the description of five usage scenarios for applying ReleasePlanner™. A case study project is given in Section 6. It is intended to illustrate the introduced concepts and procedures of both EVOLVE* and ReleasePlanner™. Practical experience in using the tool is reported in Section 7. Finally a summary and an outlook are given in Section 8.

## 2. Model Building and Problem Statement

### 2.1 Informal Problem Statement and Relation to other Disciplines

The requirements engineering process is a decision-rich problem solving activity [Aurum & Wohlin 03]. One of the most prominent issues involved in incremental software development is to decide upon the most promising software release plans while taking into account diverse qualitative and quantitative project data. This is called release planning. The input for the release planning process is a set of features that are evolving due to changing user requirements and better problem understanding. Despite the obvious importance of the problem in current incremental and evolutionary development, it is poorly studied in the literature.

Release planning considers stakeholder priorities and different types of constraints. The output of the release planning process is a set of candidate assignments of features to releases. They are supposed to represent a good balance between stakeholder priorities and the shortage of resources. In each release, all the features are executed following one of the existing software development paradigms including analysis, system design, detailed design, implementation, component testing, system testing, and user testing. All the features are inputted into this process. As a result, a usable (release) product is provided. This fundamental procedure of planning and development of releases is illustrated in Figure 1. Release planning assigns features to release options 1 (dark grey), 2 (grey) or 3 (white). Within each release development cycles, all features are passing the stages of a software development cycle. This cycle includes verification and validation activities at the different product stages (requirement, system design, component design, and code). At the end of this process, a release product is delivered. This principle can be easily extended to planning of more than two releases ahead.

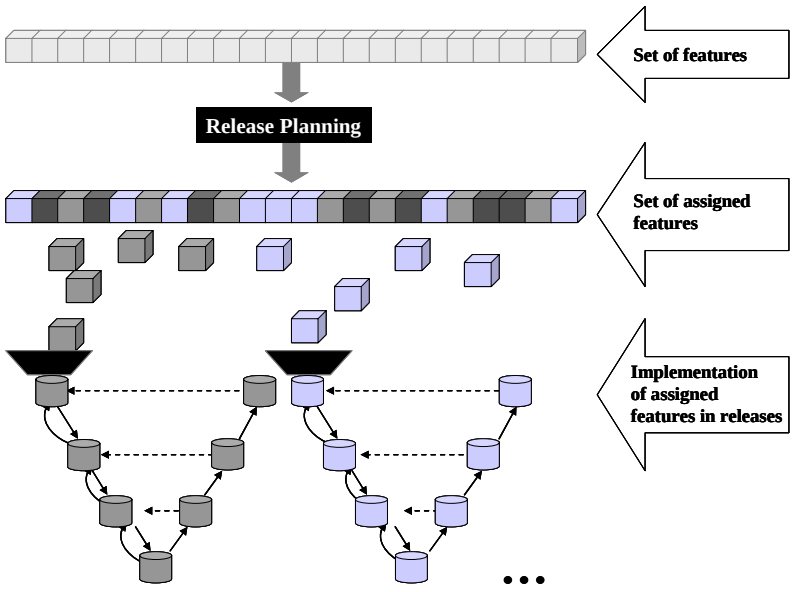


**Figure 1. Process of planning and development of releases.**

Without any technological, resource, risk and financial constraints, all the features could be implemented in one release. However, the existence of all the constraints implies the questions: what comes first and why? The goal of release planning is to account for all these factors and to come up with suggestions for the most satisfactory release plans. There are two fundamental types of release planning problems: (i) release planning with pre-determined time interval for implementation, and (ii) planning with flexible intervals. In the second problem, you also decide about the length of the interval to implement all the assigned features. In this article, we will focus on release planning with pre-determined intervals.

Software release planning adds to two well-established disciplines of (incremental) software development: (i) requirements management, especially requirements prioritization, and (ii) software project planning and management. Defining the commonalities and differences between them helps to better understand release planning. Requirements management is the process of identifying, documenting, communicating, tracking and managing project requirements as well as changes to those requirements. As requirements are changing, or are becoming better understood, or new requirements are arising, requirements management is an ongoing activity.

Requirements prioritization is trying to determine the different degrees of priority. The problem of still delivering a large amount of features that are never used, and vice versa, not delivering those that are required, has (among others) to do with a lack of understanding and prioritization. As a feature has different relevant attributes (such as its functionality, inherent risk, effort of implementation) that contribute to the final judgement, requirements prioritization is a multi-attributive decision problem. Practically, most emphasis is on the provided functionality of the feature. Specifically, requirements prioritization is also a multi-person (multi-criteria) decision problem, as the prioritization is typically performed in a team-session. There is no clear description on how the different and conflicting opinions are actually negotiated. Software release planning as formulated later in this article extends prioritization in five directions:

- Release planning is based on a set of (representative and most important) stakeholders which can input priorities from a remote place (don't need to attend a physical meeting) via access to the web.
- There is a formal procedure to balance all the stakeholder priorities to determine an overall prioritization result. As part of that, the degree of importance of the different stakeholders can be varied.
- Release planning takes into account (estimates of) the implementation effort. In addition, it can also handle constraints related to risk, money, or technological dependencies.
- Release planning considers the time of implementation by assigning features to releases.
- The formal procedure of release planning is able to consider different criteria (urgency, importance) and to bring them together in a balanced way.

Software project planning and management is the art of balancing competing objectives, managing risk, and overcoming constraints to successfully deliver a product that meets the needs of both customers and the users. The planning stage is based on decisions about which features should be implemented in which release – as provided by release planning. In other words, release planning results are the input for the more detailed project planning and project management.

## 2.2 Requirements, Features and Constraints

Release planning is based on an evolving set of features and requirements. The number of releases to be considered in advance may vary from case to case. We use "option k" for the assignment of a feature to release k. Because of the high degree of requirements volatility, it does not make sense to plan too many releases in advance. For this article (and without loss of generality), we will only consider two releases in advance. This is considered to be a good compromise of looking into the future while accepting the uncertainty and volatility of features. Consequently, as a result of release planning, each feature is assigned to exactly one of three possible cases:

- next release (option 1),
- next but one release (option 2), or
- postponed or not (yet) considered for implementation (option 3).

Let $F = \{f_1,\ldots, f_n\}$ be the set of features to be assigned to releases. Whenever applicable without ambiguity, $\{1,2,\ldots,n\}$ is used instead. As introduced above, release planning is distributing F into three categories: "next release" (option 1), "next

but one release" (option 2), and "not yet decided" (option 3). Consequently, a release plan is characterized by a vector x of decision variables

(1) $x = (x(1),x(2),\ldots,x(n))$ with $x(j) = k$ if feature j is assigned to option k

Assignment of features to releases can't be done without considering the different types of dependencies between features. These dependencies are represented by relations defined on the product set $F*F$ of F. In [Carlshamre et al. 01], six different types of requirements dependencies were analyzed. From a release planning perspective, we formulate them below as six types of dependency between features. Some of them are illustrated by feature dependencies of a text processing system.

**Type 1**

i AND j if feature i requires feature j to function and vice versa.

If two features are in Type 1 relationship then they must belong to the same release.

**Type 2**

i FREQ j if feature i requires feature j to function, but not vice versa.

If two features are in Type 2 relationship then i should not be implemented in an earlier release than j.

**Type 3**

i TREQ j if implementation of feature i requires implementation of feature j.

If two features are in Type 3 relationship then i should be implemented in the same release as j.

**Type 4**

If two features are in Type 4 relationship then the value of the two in combination is different from the additive value (non-additive value function) when applied in isolation. The feature '

**Type 5**

If two features are in Type 5 relationship then the effort of the two in combination is different from the additive value (non-additive effort function).

**Type 6**

For the sake of simplicity, we assume that dependencies of types 4 and 5 are handled by synthesizing these features into a new (integrated) one. To model the remaining types of dependencies, we introduce a directed graph $G(R) = (V(F), A(F))$ with a set of vertices V(F) and a set of arcs A(F). Feature $i_i$ from a set of features F is represented by vertex set $i \in V(F)$. The set A(F) of arcs is defined from dependencies of Type 1 to 3 as follows:

Type 1: i AND j implies $(i,j) \in A(F)$ and $(j,i) \in A(F)$

Type 2: i FREQ j implies $(i,j) \in A(F)$

Type 3: i TREQ j implies $(i,j) \in A(F)$

With the graph $G(F) = (V(F), A(F))$, we can formulate dependency constraints as

(2) $(i,j) \in A(F)$ and $(j,i) \in A(F)$ for type 1 dependency and $(i,j) \in A(F)$ for type 2 and type 3 dependency

Release planning is impacted by a variety of constraints. These constraints can be related to different aspects such as effort, risk, or budget. The effort to implement a requirement is hard to estimate. For the easiest case, we assume an effort function, effort: $\Re \rightarrow R^+$ assigning to each feature an estimated effort for its implementation. As this effort is typically

hard to predict, we assume that it is based on estimates such as the optimistic, the pessimistic and the most likely effort. For the sake of simplicity, we further assume an additive function for the implementation of set $A \subset F$.

Resources are an essential asset for planning of releases. Consideration of just 'effort' is just a rough approximation and does not consider individual types of resources. For each release, we consider all necessary resources for performing the tasks assigned to this release. Definition of the actual resource types is done by the project manager. Example resource types are:

- Requirements specification
- Analysis and design
- Implementation
- Testing
- Money
- Risk

Let's assume that R different resource types are considered to realize the different features. Not all resource types are necessarily needed for all features. We further assume resource capacity bounds Resource_Bound(k,r) for the two releases to be planned and for all types of resources r = 1,…,R. All feasible release plans have to fulfill the resource capacity constraints for release options 1 and 2, e.g.,

(3) $\text{Resource}(k,r,x) := \sum_{x(i)=k} \text{resource}(i,r) \leq \text{Resource\_Bound}(k,r)$

for releases k = 1,2 and for all resource types r = 1,…,R

In the same way, we can perform a risk evaluation for each feature. Risk estimation is used to address all the inherent uncertainty associated with the implementation of a certain feature. We employ a risk score as an abstraction of all risks associated with a given feature. These risks may refer to any event that potentially might negatively affect schedule, cost or quality in the final project results [Ruhe & Greer 03].

For each feature i, 'risk' is an interval scaled function, risk: $F \rightarrow [0,1)$, where '0' means no risk at all and '1' stands for the highest risk. In what follows we assume that the risk assessment is done by expert judgment. We further assume, that the risk is independent from the assigned release. The objective of risk balancing is to avoid a concentration of too many risky features in the same release. The risk per release is supposed to be additive, and Risk_Bound(k) denotes the upper bound for the acceptable risk for options 1 and 2. This leads to constraints

(4) $\text{Risk}(k,x) := \sum_{x(i)=k} \text{risk}(i) \leq \text{Risk\_Bound}(k)$ for k = 1,2

In some cases, financial constraints are important as well (or even the most important constraint). From a purely monetary perspective, we assume an estimated financial effort required to realize feature i. For each feature i, 'finance' is an interval scaled function, finance: $F \rightarrow R^+$ assigning to each feature an estimated amount of money for its implementation. As there is an available financial budget Finance_Bound(k) for both the next two releases under consideration, all feasible release plans have to fulfill

(5) $\text{Finance}(k,x) := \sum_{x(i)=k} \text{finance}(i) \leq \text{Finance\_Bound}(k)$ for k = 1,2

With constraints as introduced above, we are able to define feasibility of release plans. A release plan x is called feasible if it fulfills all the model constraints (2) - (5). The set of all feasible release plans is denoted by X.

## 2.3 Stakeholder Priorities

### 2.3.1 Stakeholder

One of the challenges of software development is to involve stakeholders in the requirements engineering process. System stakeholders in the area of software engineering are defined as "people or organizations who will be affected by

the system and who have a direct or indirect influence on the system requirements" [Kotonya & Sommerville 99]. An approach for identification of stakeholders is addressed in [Sharp et. al 99].

Effectively solving the problem of release planning involves satisfying the needs of a diverse group of stakeholders. Stakeholder' examples are: user (novice, advanced, expert or other classifications of users), manager (project, product), developer, or sales representatives.

We assume q different stakeholders abbreviated by $S_1$, $S_2$,…,$S_q$. Each stakeholder $S_p$ is assigned a relative importance $\lambda_p \in (0,1)$. The relative importance of all involved stakeholders is typically assigned by the project or product manager. If it is difficult to actually determine these weights, pair-wise comparison using the analytic hierarchy process [Saaty 80] can be used as a support. We assume that stakeholder weights are normalized to one, i.e.,

(6) $\Sigma_{p=1,\ldots,q} \lambda_p = 1$

### 2.3.2 Prioritization

The increasing focus on value creation as a result of software development implies the question of the impact on value for the different features or requirements. Typically, there are different and conflicting priorities between different (groups of) stakeholders. To determine the most attractive feature and product portfolio's, priorities have to be evaluated to the best knowledge available. There are different ways to evaluate the 'priority' of a feature from a stakeholder perspective. For our purposes, we consider two dimensions of priority: a value-based, and an urgency-based prioritization. Value addresses the assumed impact on the value of the final product. Value here is considered to be independent of time. Urgency more addresses the time-to-market aspect, maybe, to reflect market needs and competitor analysis information. Intuitively, what we are trying to achieve is to assign features of high value and high urgency to first releases. We define two attributes for priority evaluation:

### 2.3.3 Value

Value-based software engineering can help to identify a process in which value related decisions can be integrated in software engineering practices. [Boehm & Huang 03] state that we can no longer afford to follow a value-neutral approach where

- software engineers treat every requirement, feature, use case, object, defect or other artefacts as of equal value;
- methods and practices are largely logical activities not primarily taking into account the creation of value;
- software engineers use earned-value systems to track project cost, schedule, but not stakeholder or business value;
- concerns are separated from software engineers' turning requirements into verified goals, and
- setting goals for improving productivity or correctness independent of stakeholder value considerations.

There is no easy and crisp definition of value. It can be (i) a fair return or equivalent in goods, services, or money, or (ii) the monetary worth of something, or (iii) relative worth, utility or importance. Without being more precise about the concrete meaning, we expressed 'value' by a nine-point (ordinal) scale with increasing order of value corresponding to increasing value-based priority. The judgment of stakeholder p with regard to feature i is denoted by value(p,i) where value(p,i) $\in \{1,2,\ldots,9\}$ represents the perceived value of the feature i for stakeholder p. We are using a nine-point scale of measurement which could be replaced by another (e.g., five-point) scale in dependence of the degree of knowledge about the subject. The higher the number, the higher the perceived value. As a guideline, we define

(7) value(p,i) = 1 if feature i is of very low value

(8) value(p,i) = 3 if feature i is of low value

(9) value(p,i) = 5 if feature i is of moderate value

(10) value(p,i) = 7 if feature i is of high value

(11) value(p,i) = 9 if feature i is of extremely high value

### 2.3.4 Urgency

We have introduced the three options on how to assign a feature to releases. Urgency addresses the degree of satisfaction with these three possible options from the individual stakeholder perspective. Urgency can be motivated by time-to-market of certain product features. For each feature i, the stakeholder p is asked to represent his/her satisfaction with the situation that feature i is assigned to option k (k=1,2,3). His/her judgment is expressed by $sat(p,i,k) \in \{1,\ldots,9\}$ with

(12) $Sat(p,i) = (sat(p,i,1), sat(p,i,2), sat(p,i,3))$ for all features i and all stakeholder p

(13) $\sum_{k=1,2,3} sat(p,i,k) = 9$ for all features i and all stakeholder p

The higher the number of votes (s)he assigns to k, the more satisfied (s)he would be if the feature is put in the respective option. As a consequence, a urgency vote $Sat(p,i) = (9,0,0)$ expresses highest possible urgency assigned by stakeholder p to feature i. Similarly, $Sat(p,i) = (5,4,0)$ describes a slight preference for assigning feature i to release 1 instead of release 2. Finally, $Sat(p,i) = (0,0,9)$ describes a degree of uncertainty or non-importance with respect to feature i from the perspective of stakeholder p.

## 2.4 Objectives and Formal Problem Statement

The question of what actually constitutes a good release plan needs careful consideration. Intuitively, we are expecting most valued and most urgent features first. However, 'most valued' and 'most urgent' might refer to different features for different stakeholders. The user is expecting features that he or she would need first to get started. But there are different types of users such as novice, advanced and expert users having different types of expectations and preferences.

Potentially, there is a great variety of formally stated objective functions. Some example functions are discussed in [Ruhe & An-Ngo 04]. We propose a function combining the individual stakeholder evaluations from the perspective of value and urgency in a multiplicative way. For each feature i, we introduce the weighted average priority WAP(i,k) reflecting the product of the value and the urgency of stakeholders with respect to a fixed feature and a fixed option of assignment to releases.

Typically, not necessarily all stakeholders are able or comfortable to give their evaluation related to all features. A stakeholder p is called active with respect to feature i, if (s)he was assigned to provide an evaluation (and is called passive otherwise). The set of active stakeholders with respect to feature I is denoted by P(i). For determining WAP(i,k), the sum of all individual weighted products is divided by the sum of all active stakeholder weights.

Further flexibility is introduced by the possibility to vary importance of stakeholder weights $\lambda_p$ and to weight the importance $\xi_1$ and $\xi_2$ of release option 1 and release option 2, respectively. This yields the objective function F(x) defined as

(14) $F(x) = \xi_1 \sum_{x(i)=1} WAP(i,1) + \xi_2 \sum_{x(i)=2} WAP(i,2)$ with

(15) $WAP(i,k) := [\sum_{p \in P(i)} \lambda_p \cdot value(p,i) \cdot sat(p,i,k)] / [\sum_{p \in P(i)} \lambda_p]$ for all features i and k = 1 and 2.

For a fixed vector $\lambda$ of stakeholder weights and a fixed vector of bounds, the release-planning problem $RP(\lambda,\xi, Bound)$ becomes

(16) Maximize $F(x, \xi, \lambda, Bound)$ subject to $x \in X$.

(16) represents a specialized integer programming problem. As described in [Ngo-The & Ruhe 04], we will generate a set of alternative solutions having the additional property of being maximally diversified in terms of the structural distance between all pairs of involved solutions. This solution set is typically small in terms of the number of solutions involved. Decision-Making under incompleteness and uncertainty is supported by providing a qualified set of solution having the property of being maximally diversified. This increases the understanding of the possible range of solutions and improves

the chance to offer a solution finally accepted by the decision-maker. From a solution set generated this way, the decision-maker can finally choose his or her most preferred solutions

To perform what-if analysis or re-planning, a sequence $\{RP(\lambda^i,\xi^i, Bound^i)\}_{i=1,2,..}$ of problems with varying parameters $\lambda^i$, $\xi^i$ and $Bound^i$ is typically solved. Risk mitigation by investigating the impact of different risk levels for both releases is an important application of that. Another one comprises resource planning with changing levels of resource capacities.

## 3. Difficulties with Release Planning

[Carlshamre 02] characterizes the release planning as "wicked" in the sense defined by [Rittel & Webber 84]. That means that the objective is "to maximize the benefit", but it is difficult to give a measurable definition of "benefit". Wicked problems have no stopping rule in its solution procedure. The underlying model is "evolving": the more we study the problem, the more sophisticated the model becomes. Wicked problems have better or worse solutions, but no optimal one. Although we are approximating the reality, implicit and tacit judgment and knowledge will always influence the actual decisions. As a consequence of all these difficulties, we propose to rely on the synergy between computational strength and the experience and intelligence of the human decision maker as proposed by the paradigm of software engineering decision support [Ruhe 03].

Release planning is a very complex problem including different stakeholder perspectives, competing objectives and different types of constraints. Release planning is impacted by a huge number of inherent constraints. Most of the features are not independent from each other. Typically, there are precedence and/or coupling constraints between them that have to be satisfied. Furthermore, effort, resource, and budget constraints have to be fulfilled for each release. The overall goal is to find a relatively small set of "most promising" release plans such that the overall value and the degree of satisfaction of all the different stakeholders are maximized. The topic of investigation is uncertain and incomplete in its nature [Ruhe & Ngo-The 04]:

- **Features are not well specified and understood:** There is usually no formal way to describe the features and requirements. Non-standard format of feature specification often leads to incomplete descriptions and makes it harder for stakeholders to properly understand and evaluate features and requirements.
- **Stakeholder involvement**: In most cases, stakeholders are not sufficiently involved in the planning process. This is especially true for the final users of the system. Often, stakeholders are unsure why certain plans were suggested. In the case of conflicting priorities, knowing the details of compromises and why they were made would be useful. All these issues add to the complexity of the problem at hand and if not handled properly, they create a huge possibility for project failures
- **Change of features and requirements** and other problem parameters: Features and requirements always change as the project progresses. If a large number of features increase the complexity of the project, their dynamic nature can pose another challenge. Other parameters such as the number of stakeholders, their priorities, etc., also change with time - adding to the overall complexity.
- **Size and complexity** of the problem: Size and complexity are major problems for project managers when choosing release plans - some projects may have hundreds or even thousands of features. The size and complexity of the problem (known to be NP-complete), and the tendency for not involving all of the contributing factors, makes the problem prohibitively difficult to solve by individual judgment or trial and error type methods.
- **Uncertainty of data:** Meaningful data for release planning are hard to gather and/or uncertain. Specifically, estimates of the available effort, dependencies of features, and definition of preferences from the perspective of involved stakeholders are difficult to gauge.
- **Availability of data:** Different types of information are necessary for actually conducting release planning. Some of the required data are available from other information sources within the organization. Ideally, release planning is incorporated into existing Enterprise Resource Planning or other organizational information systems.
- **Constraints:** A project manager has to consider various constraints while allocating the features and requirements to various releases. Most frequently, these constraints are related to resources, schedule, budget or effort.

- **Unclear objectives:** 'Good' release plans are hard to define at the beginning. There are competing objectives such as cost and benefit, time and quality, and it is unclear which target level should be achieved.
- **Efficiency and effectiveness** of release planning: Release plans have to be updated frequently due to changing project and organizational parameters. Ad hoc methods help determine solutions but are far behind objective demands.
- **Tool support:** Currently, only general-purpose tools for features management are available. Most of them do not focus on the characteristics of release planning.

## 4. Solution Methods and Techniques

### 4.1 Requirements Prioritization

Prioritization in general answers the questions to classify objects according to their importance. This does not necessarily imply a complete ranking of all objects, but at least an assignment to classes like 'Extremely important', 'important', or 'of moderate importance'. Different scales are applicable according to the underlying degree of knowledge you have about the objects. Prioritization always assumes one or a collection of criteria to actually perform the process.

[Karlsson et al. 98] characterize a requirements prioritization session by three consecutive stages:

- Preparation: Structuring of the requirements according to the principles of the method to be applied. Provide all information available.
- Execution: Agreement on criteria between all team members. Decision makers do the actual prioritization with all the information available. In general, this step needs negotiation and re-iteration.
- Presentation: Final presentation of results to those involved in the process.

There are a number of existing approaches to requirements prioritization. The most important ones have been studied and compared in [Karlsson et al. 98]. Among them are the analytic hierarchy process (AHP), binary search tree creation, greedy-type algorithms and other sorting-based methods. As a result of their evaluation, they have found out that AHP is the most promising approach. Most of those algorithms need $O(n^2)$ comparisons between the n requirements. This effort required soon becomes prohibitive for larger number of requirements. In addition to that, none of the mentioned algorithms takes into account different stakeholder perspectives.

The Analytic Hierarchy Process (AHP) [Saaty 80] is a systematic approach to elicit implicit preferences between different involved attributes. For the purpose of this investigation, AHP is applied to determine the importance of the various stakeholders from a business perspective. In addition, it is used to prioritize the different classes of requirements from the perspective of each stakeholder. The two preference schemata are combined to judge rank the importance of the different classes of requirements for the final business value of the software product. AHP assumes that the problem under investigation can be structured as an attributive hierarchy with at least three levels. At the first level, the overall goal is described. The second level is to describe the different competing criteria that are refining the overall goal of level 1. Finally, the third level is devoted to be used for the selection from competing alternatives. At each level of the hierarchy, a decision-maker performs a pair-wise comparison of actions or criteria. The result is an assessment of their contributions to each of the higher level nodes to which they are linked. This pair-wise comparison involves preference ratios (for actions) or importance ratios (for criteria).

Priority decisions become more complicated in the presence of stakeholders having different relative importance and different preferences. It becomes very hard in the presence of constraints about sequencing and coupling of requirements in various releases. Even more so if you are taking into account different resource types necessary for the implementation of requirements. None of the methods mentioned above can be applied under these circumstances.

### 4.2 Planning Games

Amongst the informal approaches claiming to handle release-planning, one of the most well known ones is 'planning games' as used in agile development [Cockburn02]. The goal of this approach is to deliver maximum value to the customer in least time possible. In half to one day long sessions, customers write story cards describing the features they

want, while developers assign their estimates to those features. The customers then choose the most promising story cards for the next release by either setting a release date and adding the cards until the estimated total matches the release date, or selecting the highest value cards first and setting the release date based on the estimates given on them.

This simplistic approach works well in smaller projects [Amandeep et al. 04]. However, as the size and the complexity of the projects increases, the decisions involved in release planning become very complex. Various factors come into play, such as the presence of stakeholders having different relative importance and different preferences, the presence of constraints about sequencing and coupling of requirements in various releases, and the need to take into account resource allocation issues for implementing the requirements. Considering problems involving several hundreds of requirements and large number of widely scattered stakeholders, it becomes very hard to find appropriate solutions without intelligent support tools. The goal of such support is to account for all these factors in order to come up with a set of most promising release plans.

## 5. Intelligent Decision Support for Release Planning

### 5.1 Evolutionary Solution Approach EVOLVE*

We have observed that the problem of release planning is extremely difficult because of its inherent uncertainty, size and complexity. It is unrealistic to expect that this problem can be completely solved through one iteration. Instead, our strategy is to try to gradually reduce the size and complexity of the problem and to increase the validity of the underlying model. Finally, we get a set of candidate solutions with reasonable size to be considered in depth by the decision makers.

The overall architecture of EVOLVE* is designed as an iterative and evolutionary procedure mediating between the real world problem of software release planning, the available tools of computational intelligence for handling explicit knowledge and crisp data, and the involvement of human intelligence for tackling tacit knowledge and fuzzy data. This is illustrated in Figure 2.

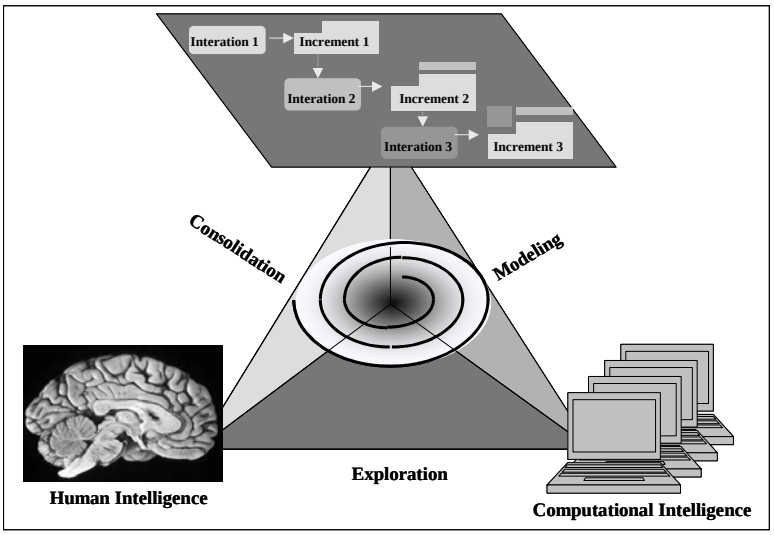


**Figure 2: EVOLVE* as a mediator between real problem world, computational intelligence based tools, and human intelligence.**

The spiral curve describes the performance of EVOLVE*. At all iterations, three phases are passed:

- **Phase 1** - Modeling: Formal description of the (changing) real world to make it suitable for computational intelligence based solution techniques. This includes the definition of all decision variables, as well as their dependencies and constraints, and the description of what is, or contributes to, the “goodness” of a solution. Other data, such as stakeholder evaluation of all features, are also part of modeling.

- **Phase 2** - Exploration: Application of specialized optimization techniques to explore the solution space, to generate solution alternatives and to evaluate them according to the given criteria.
- **Phase 3** - Consolidation: Human decision maker is invited to investigate current solution alternatives. This contributes to the understanding of the problem and results in modifying parts of the underlying model or in some local decisions (e.g., pre-assigning some features to a release). Typically, these decisions reduce the size and complexity of the problem for the next iteration.

## 5.2 Overview ReleasePlanner™

ReleasePlanner™ is a web–based tool suite (see www.releaseplanner.com for further details) that provides a flexible and web-based tool support for assigning requirements or features to releases such that most important risk, resource, and budget constraints are fulfilled. The tool was developed at the Laboratory for Software Engineering Decision Support http://www.seng-decisionsupport.ucalgary.ca at the University of Calgary. It can be used in different usage scenarios and aims in providing intelligent decision support for any kind of prioritization and iterative product development. It addresses the wicked character of the problem by an approach integrating computational and human intelligence.

ReleasePlanner™ is based on a solution approach called EVOLVE* that combines the strength of specialized optimization algorithms with the flexibility of an iterative solution method. At all iterations, a specialized optimization algorithm are applied to determine the most promising solution alternatives. Main features of the tool are:

- web-based tool that allows input and interaction even with remote stakeholders. This results in a much better chance to actually achieve high customer satisfaction with the products to be developed.
- Generation of a set of most promising alternative solutions. This allows to better approach inherent uncertainty and incompleteness of data as well as allowing decision-making including implicit human preferences.
- Compatibility with commercial tools addressing features management (as a prerequisite for using release planning) and project management (for actually planning and controlling the performance of the plan).
- Computational strength of its core optimization algorithms capable of solving even large scale problems in seconds.
- Applicability to a variety of usage scenarios. For further details on that, please compare Section 5.2.
- Modeling support is offered. As validated models are a mandatory prerequisite for meaningful results, different objectives (value maximization, optimal customer satisfaction), different schema of evaluation, and different types of constraints (effort, resource, precedence, finance, risk) can be included.
- Flexibility in granularity. ReleasePlanner™ is able to work with different levels of abstraction. The lowest level is that of individual requirements, the highest level is (groups of) features.
- Supports scenario-based what-if analysis and re-planning
- Portability: The tool can be used for different platforms and operating systems.

## 5.3 Usage Scenarios

The decision support platform is widely applicable in planning and prioritization of software and hardware products and services. Releases are known to be new versions of an evolving product. However, the idea of a release is not restricted to this, but can be applied to any type of periodic development where a release would correspond to a milestone or to an annual or quarterly time period. The special case of one release known as prioritization is of even larger applicability wherever competing items have been selected under additional constraints

In what follows we will describe five usage scenarios for planning software releases. For each scenario, some background and the scenario output are given. For all scenarios, we assume an instance of the release planning problem (16) as an input. The overall planning information is secured and is only visible by the product (or project) manager. For all involved stakeholder, individual access to information is provided and controlled by the project manager.

Mismatch of customer satisfaction still with the functionality of the delivered software is one of the main reasons that software projects fail [Standish Group 2002]. To achieve better customer satisfaction, their early involvement is crucial. In globally performing companies, stakeholders (including the different types of customer) are distributed all over the

world. Using ReleasePlanner™, they can get easily access to the system to provide their urgency and value evaluation of the features assigned to them. This is a commonality for all the scenarios described in the following.

**Background**

Strategic planning (or road-mapping) addresses assignment of features to releases on a strategic level. The time interval of planning is typical several months. Preferably, total effort and risk constraints are considered to determine alternative release plans. The goal is to find an optimal balance between competing stakeholder priorities and bottleneck resources. Re-planning is applied either for assumed or actual changes of the problem parameters. Another direction of re-planning is to understand what has to be provided to achieve certain goals.

This scenario could be supported by any kind of collaborative environment. In the easiest case, this could be a teleconference. The product or project manager guides and moderates the process. The intelligence of the tool is the backbone for making proposals on how to balance the conflicting stakeholder interests.

**Output**

- Generation of a set of most promising alternative release plans
- Comparison between strategic planning alternatives (maximally diversified)
- Reporting of complete planning information for further discussion and refinement of problem parameters
- Trade-off analysis with respect to resource consumption, risk, and overall value creation

### 5.3.5 Operational Planning and Re-Planning

**Background**

Operational planning is a refinement of strategic planning. For the next release, a sequence of releases is defined for realizing the predefined features. Planning is devoted to main tasks for realizing the individual requirements. The planning is done for shorter time frames (weeks) resulting in definition of which tasks are performed in which release. As for strategic planning, this scenario is aimed to better understand the impact of the different problem parameters.

**Output**

- Best operational planning alternatives refining strategic plan
- Evaluation of operational feasibility of proposed strategic plan
- Comparison between operational planning alternatives (maximally diversified)
- Reporting of complete planning information for further discussion and refinement of problem parameters
- Trade-off analysis with respect to resource consumption, risk, and overall value creation

### 5.3.6 Resource Planning and Risk Mitigation

**Background**

This scenario emphasises planning in terms of core resources, budgets and in terms of the anticipated risks of the features. Scenario playing here means to understand the impact of resource and budget shortages. In the same way, a comparison between the values achievable for different levels of accepted risks within a release is studied.

**Output**

Optimal strategies for resource planning and risk mitigation in dependence of varying project settings. In case of changed or added features, re-planning can easily be performed.

### 5.3.7 Planning across Projects

**Background**

This scenario is considering any of the questions discussed so far with the only difference that the planning includes not only one product or project, but several ones. Organizations typically have more than one project running, and all the planning has to consider the proper balancing between them.

**Output**

The results are analogously to the ones from scenarios the first three scenarios with the extension of being applicable to the whole set of projects (instead of just one).

### 5.3.8 Synchronization of Releases

**Background**

In the case of a product that is composed out of individual sub-products, release planning can be applied to different parts separately. For embedded software, these parts could be related to hardware, middle-ware and software components. In order to release a version of the whole product, all the releases of the sub-product have to be synchronized. Non-synchronized release plans would result in a situation where the delivery of the product is delayed whenever one of its inherent parts is delayed.

**Output**

Same principal output as above. Synchronized release plans provide optimal performance for the integrated product. All the release plans of the individual sub-products are adapted to achieve final synchronization.

## 6. Case Study

To illustrate solution approach EVOLVE* as presented above, we describe a case study project taken from the telecommunication domain. The usage scenario is “Strategic Planning and Re-Planning”. To show the different stages of solution approach EVOVLE* and its implementation in ReleasePlanner, we perform two full iterations with all the three phases therein. This is fully in line with the paradigm of decision support aiming in the generation of most promising solution alternatives for the final human decision making. Because of the uncertainty of the data and the incompleteness of the problem understanding, this result can not be expected from performing only a single iteration without incorporating the human feedback from analyzing a first set of alternative solutions.

The trial strategic project encompasses 25 features clustered into five groups. The planning process is for the next two annual releases of the company XYZ’s product development. Five out of the 23 features have been already pre-assigned to the first respectively second release. This is considered to be a predetermined decision outside the scope of the considered planning process.

Different (bottleneck) resources are required to implement the proposed features. Table 1 gives the list of all features and their estimated effort consumptions. In addition to that, the capacity bound for both releases are given for all resource types. The different resource types are:

- BTS Software Development
- BTS Hardware Development
- BSC/BSM Software Development
- MTX Software Development
- Testers
- Documentation
- Capital Requirement ($k)
- Risk (1 low risk, 10 high risk)

Table 1. List of all features and their estimated resource consumptions.

| ID/ Group | Feature | BTS Software Development | BTS Hardware Development | BSC/BSM Software Development | MTX Software Development | Testers | Documentation | Capital Requirement ($k) | Risk (1 low risk, 10 high risk) |
|---|---|---|---|---|---|---|---|---|---|
| 1/A | 16 sector, 12 carrier BTS | 480 | 500 | 150 | 200 | 400 | 400 | 1500 | 10 |
| 2/A | China Feature 1 | 50 | 0 | 250 | 140 | 200 | 60 | 500 | 3 |
| 3/A | China Feature 2 | 60 | 10 | 120 | 120 | 190 | 40 | 200 | 5 |
| 4/A | China Feature 3 | 75 | 75 | 300 | 120 | 450 | 50 | 500 | 5 |
| 5/A | China Feature 4 | 0 | 0 | 100 | 450 | 400 | 50 | 0 | 3 |
| 6/A | China Feature 5 | 250 | 100 | 400 | 400 | 400 | 50 | 300 | 5 |
| 7/B | Common Feature 01 | 100 | 0 | 250 | 100 | 200 | 0 | 50 | 5 |
| 8/B | Common Feature 02 | 0 | 0 | 100 | 250 | 150 | 50 | 0 | 4 |
| 9/B | Common Feature 03 | 200 | 0 | 150 | 0 | 100 | 20 | 0 | 4 |
| 10/B | Common Feature 04 | 100 | 0 | 300 | 200 | 200 | 30 | 50 | 5 |
| 11/C | Cost Reduction of Transceiver | 150 | 200 | 120 | 0 | 200 | 60 | 1000 | 3 |
| 12/F | Expand Memory on BTS Controller | 75 | 120 | 10 | 0 | 75 | 20 | 200 | 2 |
| 13/F | FCC Out-of-Band Emissions | 400 | 120 | 100 | 0 | 200 | 10 | 200 | 4 |
| 14/E | India BTS variant | 575 | 420 | 400 | 200 | 250 | 200 | 750 | 6 |
| 15/E | India Market Entry Feature 1 | 200 | 100 | 250 | 250 | 250 | 100 | 500 | 4 |
| 16/E | India Market Entry Feature 2 | 0 | 0 | 300 | 250 | 250 | 100 | 300 | 6 |
| 17/E | India Market Entry Feature 3 | 100 | 100 | 150 | 100 | 300 | 25 | 1200 | 9 |
| 18/C | Next Generation BTS | 450 | 350 | 375 | 125 | 500 | 200 | 150 | 8 |
| 19/D | Pole Mount Packaging | 400 | 180 | 300 | 50 | 400 | 150 | 500 | 4 |
| 20/F | Software Quality Initiative | 450 | 0 | 100 | 50 | 400 | 5 | 0 | 3 |
| 21/D | USEast Inc. Feature 1 | 100 | 0 | 500 | 200 | 400 | 100 | 0 | 2 |
| 22/D | USEast Inc. Feature 2 | 200 | 0 | 400 | 250 | 250 | 50 | 25 | 4 |
| 23/D | USEast Inc. Feature 3 | 400 | 0 | 600 | 500 | 500 | 200 | 100 | 4 |
| 24/D | USEast Inc. Feature 4 | 150 | 0 | 400 | 125 | 400 | 150 | 1000 | 8 |
| 25/D | USEast Inc. Feature 5 | 75 | 180 | 225 | 225 | 300 | 60 | 750 | 6 |
| Bounds Release 1 | | 3000 | 1200 | 3600 | 2200 | 4000 | 1000 | 6000 | 120 |
| Bounds Release 2 | | 3000 | 1200 | 3600 | 2200 | 4000 | 1000 | 5000 | 120 |

There are seven stakeholders participating in the evaluation process. They represent different types of customer organizations and different roles in the development process. Each stakeholder $S_p$ is required to give his/her judgment with respect to feature i by value(p,i) in the scale of 1 (lowest) to 9 (highest) as described in (7) – (11). The stakeholders are:

- USWest Inc. ($S_{1,}$ with weight of importance $\lambda_1 = 9$)
- Technical Director ($S_2$ with weight of importance $\lambda_2 = 8$)
- SimpleCommunication ($S_3$ with weight of importance $\lambda_3 = 6$)
- CanadaWest Inc. ($S_4$ with weight of importance $\lambda_4 = 4$)
- North communication ($S_5$ with weight of importance $\lambda_5 = 6$)
- Asia Pacific Salesman ($S_6$ with weight of importance $\lambda_6 = 3$)
- USEast Inc. ($S_7$ with weight of importance $\lambda_7 = 3$)

Features are grouped, and there are individualized access rights which stakeholder is supposed to vote which group of features. Table 2 describes the different priorities of the stakeholders in terms of their urgency and perceived value. As it can be seen from there, certain stakeholders are assigned to just vote on specific groups. That also means that the stakeholders get access only to the descriptions of the assigned features. The rationale here is that this information is highly confidential between different types of customer organizations.

For the sake of simplicity, we assume just one coupling (Type 1) and one precedence constraint (Type 2) between features are defined. USEast Inc. Feature 1 is coupled to USEast Inc. Feature 2, i.e., they depend on each other and should be delivered in the same release. Analogously, China feature 1 must precede China feature 2, e.g., it does not make any sense to deliver China feature 2 before China feature 1. These constraints are handled as hard constraints for the generation of all release plan alternatives.

Table 2. Stakeholder voting.

| ID/Group | USWest Inc. | | Tec. Director | | Simplecom. | | CanadaWest Inc. | | North comm. | | Asia Pac. Sal. | | USEast Inc. | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Urgency: | Value: | Urgency: | Value: | Urgency: | Value: | Urgency: | Value: | Urgency: | Value: | Urgency: | Value: | Urgency: | Value: |
| 1/A | (5, 4, 0) | 5 | (9, 0, 0) | 9 | (9, 0, 0) | 9 | (4, 5, 0) | 5 | No voting | | (9, 0, 0) | 9 | No voting | |
| 2/A | (9, 0, 0) | 9 | (9, 0, 0) | 9 | (8, 1, 0) | 8 | (9, 0, 0) | 9 | No voting | | (9, 0, 0) | 9 | No voting | |
| 3/A | (9, 0, 0) | 9 | (8, 1, 0) | 8 | (3, 6, 0) | 6 | (9, 0, 0) | 9 | No voting | | (7, 2, 0) | 7 | No voting | |
| 4/A | (4, 5, 0) | 4 | (0, 9, 0) | 9 | (8, 1, 0) | 8 | (0, 9, 0) | 5 | No voting | | (7, 2, 0) | 6 | No voting | |
| 5/A | (6, 3, 0) | 6 | (4, 5, 0) | 5 | (4, 5, 0) | 5 | (2, 7, 0) | 7 | No voting | | (4, 5, 0) | 5 | No voting | |
| 6/A | (7, 2, 0) | 7 | (2, 7, 0) | 6 | (2, 7, 0) | 9 | (3, 6, 0) | 5 | No voting | | (3, 6, 0) | 6 | No voting | |
| 7/B | (7, 2, 0) | 6 | (9, 0, 0) | 9 | (9, 0, 0) | 9 | (9, 0, 0) | 9 | (9, 0, 0) | 9 | (7, 2, 0) | 7 | (3, 3, 3) | 1 |
| 8/B | (7, 2, 0) | 6 | (9, 0, 0) | 9 | (9, 0, 0) | 9 | (9, 0, 0) | 9 | (5, 0, 4) | 6 | (7, 2, 0) | 6 | (3, 3, 3) | 1 |
| 9/B | (7, 2, 0) | 7 | (4, 5, 0) | 4 | (5, 4, 0) | 9 | (9, 0, 0) | 9 | (1, 6, 2) | 4 | (1, 1, 7) | 1 | (3, 3, 3) | 1 |
| 10/B | (7, 2, 0) | 7 | (5, 4, 0) | 5 | (6, 3, 0) | 9 | (5, 4, 0) | 5 | (0, 0, 9) | 1 | (1, 1, 7) | 1 | (3, 3, 3) | 1 |
| 11/C | (7, 2, 0) | 7 | (5, 4, 0) | 5 | (6, 3, 0) | 6 | (9, 0, 0) | 9 | No voting | | No voting | | No voting | |
| 12/F | (7, 2, 0) | 9 | (5, 0, 4) | 5 | (6, 3, 0) | 6 | (9, 0, 0) | 9 | No voting | | No voting | | No voting | |
| 13/F | (7, 2, 0) | 7 | (9, 0, 0) | 9 | (8, 1, 0) | 9 | (9, 0, 0) | 9 | No voting | | No voting | | No voting | |
| 14/E | (7, 2, 0) | 9 | (5, 4, 0) | 4 | (2, 7, 0) | 7 | (2, 7, 0) | 7 | No voting | | (2, 7, 0) | 7 | No voting | |
| 15/E | (7, 2, 0) | 6 | (8, 1, 0) | 8 | (2, 7, 0) | 7 | (2, 2, 5) | 2 | No voting | | (3, 6, 0) | 9 | No voting | |
| 16/E | (7, 2, 0) | 6 | (7, 2, 0) | 7 | (2, 7, 0) | 7 | (2, 2, 5) | 1 | No voting | | (3, 6, 0) | 9 | No voting | |
| 17/E | (7, 2, 0) | 7 | (7, 2, 0) | 7 | (2, 7, 0) | 7 | (2, 2, 5) | 2 | No voting | | (2, 7, 0) | 5 | No voting | |
| 18/C | (7, 2, 0) | 8 | (8, 1, 0) | 8 | (8, 1, 0) | 8 | (2, 7, 0) | 7 | No voting | | No voting | | No voting | |
| 19/D | (7, 2, 0) | 5 | (5, 4, 0) | 5 | (5, 4, 0) | 7 | (3, 6, 0) | 6 | No voting | | No voting | | (7, 2, 0) | 7 |
| 20/F | (7, 2, 0) | 9 | (2, 7, 0) | 7 | (5, 4, 0) | 5 | (9, 0, 0) | 9 | No voting | | No voting | | No voting | |
| 21/D | (7, 2, 0) | 5 | (7, 2, 0) | 7 | (9, 0, 0) | 9 | (9, 0, 0) | 8 | No voting | | No voting | | (7, 2, 0) | 9 |
| 22/D | (7, 2, 0) | 5 | (7, 2, 0) | 6 | (9, 0, 0) | 9 | (9, 0, 0) | 9 | No voting | | No voting | | (7, 2, 0) | 7 |
| 23/D | (7, 2, 0) | 5 | (6, 3, 0) | 6 | (9, 0, 0) | 9 | (9, 0, 0) | 7 | No voting | | No voting | | (8, 1, 0) | 8 |
| 24/D | (7, 2, 0) | 6 | (4, 5, 0) | 5 | (5, 4, 0) | 6 | (1, 1, 7) | 7 | No voting | | No voting | | (7, 2, 0) | 7 |
| 25/D | (7, 2, 0) | 7 | (2, 7, 0) | 7 | (4, 5, 0) | 7 | (2, 2, 5) | 5 | No voting | | No voting | | (0, 7, 2) | 9 |

With all the parameters of the project defined (as the end of Phase Modeling of EVOLVE* as described in Figure 2), the tool is now able to generate most promising solution alternatives. These alternatives are generated from a two-phased procedure. In the first step, we determine a set $X^0 \subset X$ of qualified alternative solutions. Typically, the cardinality of set $X^0$ is much lower than the cardinality of set X. In the second step, for a fixed cardinality L (determined by the decision-maker) we determine the L release plans out of $X^0$ that provide maximal diversity. The notion of diversity can be based on the "similarity" or "distance" between two solutions. Intuitively, the more two solutions are different; the more all the individual requirements are assigned to different releases. For two release plan solutions x, y from X, the degree of similarity can be represented by the Euclidean distance. The smaller the distance, the more similar the plans are.

Table 3. Structure of five alternative solutions generated in the first iteration.

| Release Plan Alternative | 16 sector, 12 carrier BTS | China Feature 1 | China Feature 2 | China Feature 3 | China Feature 4 | China Feature 5 | Common Feature 01 | Common Feature 02 | Common Feature 03 | Common Feature 04 | Cost Reduction of Transceiver | Expand Memory on BTS Controller | FCC Out-of-Band Emissions | India BTS variant | India Market Entry Feature 1 | India Market Entry Feature 2 | India Market Entry Feature 3 | Next Generation BTS | Pole Mount Packaging | Software Quality Initiative | USEast Inc. Feature 1 | USEast Inc. Feature 2 | USEast Inc. Feature 3 | USEast Inc. Feature 4 | USEast Inc. Feature 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Feature | | | | | | | | | | | | | | | | | | | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 1 | 3 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 2 | 1 | 3 | 2 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 1 |
| 2 | 3 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 2 |
| 3 | 3 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 3 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 2 |
| 4 | 2 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 3 | 2 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 3 | 2 |
| 5 | 2 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 3 | 2 | 1 | 1 | 1 | 3 | 1 | 1 | 1 | 1 | 2 | 2 |

Table 3 presents the five alternative solutions generated this way. They are supposed to be most promising with respect to the original objectives. However, because of all the inherent uncertainty of data, it is not enough to exclusively look at the

numerical values represented by the objective function performance. ReleasePlanner™ is following the decision support paradigm and offers these alternative solutions in conjunction with options to further study their structure and to compare between them.

In the consolidation phase, the human decision maker is supposed to study the appropriateness of the proposed solutions. There is the assumption that not everything can be explicitly described in the problem modeling, and that there is implicit knowledge and preferences that are influencing the choice of the final plan. We assume here, that we want to generate plans with a lower level of anticipated risk (Risk_Bound(1) = 60 respectively Risk_Bound(2) = 80). In addition, we assume an increased budget for the first release (Budget_Bound(1) = 8000). Table 4 gives the final results from this replanning.

Table 4. Structure of five alternative solutions generated in the second iteration.

| | 16 sector, 12 carrier BTS | China Feature 1 | China Feature 2 | China Feature 3 | China Feature 4 | China Feature 5 | Common Feature 01 | Common Feature 02 | Common Feature 03 | Common Feature 04 | Cost Reduction of Transceiver | Expand Memory on BTS Controller | FCC Out-of-Band Emissions | India BTS variant | India Market Entry Feature 1 | India Market Entry Feature 2 | India Market Entry Feature 3 | Next Generation BTS | Pole Mount Packaging | Software Quality Initiative | USEast Inc. Feature 1 | USEast Inc. Feature 2 | USEast Inc. Feature 3 | USEast Inc. Feature 4 | USEast Inc. Feature 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Release Plan Alternative** | **Feature** | | | | | | | | | | | | | | | | | | | | | | | | |
| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** |
| **1** | 3 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 3 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 2 |
| **2** | 3 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 2 | 2 | 1 | 3 | 1 | 1 | 1 | 1 | 2 | 2 |
| **3** | 2 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 3 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 3 | 2 |
| **4** | 3 | 1 | 1 | 2 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 2 |
| **5** | 3 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 3 |

As a final result from performing the update in the model, to perform further computations to generate a new set of most promising alternative solutions and from their human analysis one most appropriate (and best understood) solution is selected. This is a qualified solution that is close to the maximum value, but it has not necessarily to be the one with the highest numerical value but with a high degree of confidence and satisfaction of the human decision maker.

## 7. Release Planning at iGrafx

A typical scenario of performing release planning is described in (Amandeep et al. 04). Initially, iGrafx (a Product Group of Corel Inc.) releases were planned in an informal fashion using a Lotus Notes database of collated customer requirements. Records in the database were created for each distinct requirement received from customers (existing and prospective) and internal sources. In addition to title, description, and source, records included a priority ranking between 1 (high) and 4 (low). An effort field estimated the development time (in days) to complete requirements.

The Lotus Notes database was effective for providing a fast, simple, central storage of requirements available to program management (release planners) and software engineers (the developers). Lotus Notes was also good for creating a diverse set of reports and summaries to assist implementation. The planning process, however, was mostly ad hoc. Program management scanned the requirements attempting to identify the most commonly requested feature requests to add to the next release. The development team would then estimate how many requirements could be completed in the time allocated and the product plan would gradually come together. This process had several difficulties:

- Many requirement descriptions were not fully described.
- Stakeholders (other than program management) had no synchronized involvement in ranking.

- Coupling and precedence relationships between requirements were not formally defined.
- For many requirements, the effort was not defined early enough in the requirement selection process.
- Without a clear prioritization process, it was hard to achieve buy-in on the product plan.
- It was difficult to change the product plan when effort estimates changed or new “must-have” requirements arrived.

These planning deficiencies led to the discovery of ReleasePlanner™. Program management was quickly excited about the possibilities these new tools provided but software developers and other stakeholders were initially skeptical. They believed that ReleasePlanner™ would only add unnecessary complexity to the release process. This negative perception has disappeared. Now there is universal agreement among the iGrafx team that ReleasePlanner™ is improving the iGrafx product planning process and will deliver long-term value.

To date, we’ve run tests on a subset (about 250 requirements ranked by three stakeholders) of our database and the results are encouraging. For the first time, we are defining a plan that looks three releases into the future instead of only one (our past practice). In addition, we feel that the contents of each release are maximizing benefits to the stakeholders. Soon, we will define releases using the full set of 800 requirements ranked by seven stakeholders. Recently, other Corel product groups (e.g. XML Solutions and WordPerfect) have asked for information about the tool. They are keen to build on the experiences and success of the iGrafx team.

The application of ReleasePlanner™ is helping the company to define a better product plan and providing the following benefits:

- Engineers are more careful to accurately estimate implementation effort. A good estimate is now required because the effort field directly determines release contents. In addition, by defining all precedence and couplings early in the planning process, the release contents are more certain and subject to fewer revisions.
- Requirement descriptions have improved. Stakeholders will not rank requirements with poorly written descriptions or will rank them lower. Poorly described requirements will probably not land in any release.
- Stakeholders are providing earlier and more valuable feedback. Before, stakeholders were asked for advice after the initial product plan was announced. Now, they are involved earlier and are more likely to endorse final release plans.
- Program Management can more easily identify and develop release themes that can later be used by marketing.
- Customers can be given a “road-map” of future plans extending beyond the next release.

## 8. Summary and Outlook

Release planning is an important and integral part of any type of incremental product development. There is a great variety of application scenarios about when and how to apply this process. Release planning is a wicked problem and needs ongoing effort. To make it successful, an organization has to understand their priorities and constraints. In dependence of maturity and the specific needs, different scenarios with different constraints and objectives can be applied. In addition to that, the granularity of the planning process can be varied in dependence of the degree of certainty about all the data.

In this article, we have presented a formal description of release planning. This formal problem description is the cornerstone of n evolutionary solution approach that integrates human and computational intelligence in a proper way. We have further assumed that planning is always looking just two releases ahead, although the principal solution method can easily extended to other cases.

Release planning following EVOVLE* and using ReleasePlanner™ has to be evaluated in terms of the added value generated. We foresee three directions of impact: (i) Time savings for both the performing decision-maker (project or product manager) and the involved stakeholders. For both groups, time-consuming meeting can be replaced or at least reduced by using the proposed web-service; (ii) The quality of the plans becomes better by the ability to include the complete portfolio of stakeholders. This also increases the likelihood to actually achieve higher customer satisfaction for

the resulting release products; and (iii) Systematic planning addressing all important objectives and constraints allows higher probability of actually implementing the proposed features within time, budget and target quality.

The process of release planning is based on the ideas of constant learning and improvement derived from the Quality Improvement Paradigm (QIP). QIP is a methodological framework for goal-oriented systematic improvement. It was originally developed for (but is by no means restricted to) Software Engineering projects and organizations [Basili et al. 01]. Since release planning strives to learn from, and improve, projects at the organizational level, the QIP can guide the activities and goals of release planning [Amandeep et al. 04]. QIP was designed to explicitly identify improvement goals, to capture relevant knowledge, and to evaluate and systematically reuse that knowledge. Therefore, a strong goal-orientation of the entire process as well as quality control of the captured knowledge is an important built-in feature of the QIP.

Release planning does not make sense if not done in coordination with other project and process improvement efforts. Proper requirements management including the elicitation, specification, and analysis of requirements and features is an essential prerequisite for release planning. But it definitely adds value by analyzing the huge number of possible release strategies. It offers support in defining optimal or near-optimal release plans that can be taken as an input to finally decide in consideration of additional and implicit objectives and constraints.

Future research in release planning is going in different directions. Instead of having fixed release times, we could anticipate problems with flexible dates when a new releases product is issued. Furthermore, the additivity with respect to effort, value or risk is not necessarily satisfied in real-world. What we would need is solution approaches with more flexible function defined on the power set of F.

## ACKNOWLEDGEMENT

The authors would like to thank the Alberta Informatics Circle of Research Excellence (iCORE) for its financial support of this research. Many thanks are due to Des Greer, Amandeep, An Ngo-The, Omalade Saliu, Pankaj Bhawnani, Pete Garrett, Sebastian Maurice, and Michael Richter for their collaboration and stimulating discussions. Kenny Tsang, Gregory Spiers, Erik Bauld, David Goodladd, and Shahin Charkhadeh have contributed to main parts of the implementation of EVOLVE*.

## REFERENCES

[Amandeep, Ruhe & Stanford 04]
Amandeep, Ruhe G., Stanford, M., “Intelligent Support for Software Release Planning”. Proceedings PROFES’2004, Lecture Notes on Computer Science, Vol. 3009, pp. 284-262.

[Aurum & Wohlin 03]
Aurum, A., Wohlin, C., “The Fundamental Nature of Requirement Engineering Activities as a Decision-Making Process”, Information and Software Technology 2003, Vol. 45 (2003), No. 14, pp. 945-954.

[Basili, Caldiera & Rombach 01]
Basili, V., Caldiera, G., Rombach, D.: “Experience Factory”. In: J. Marciniak: Encyclopedia of Software Engineering, Volume 1, pp. 511-519, 2001.

[Beck 01]
Beck, K., “Extreme Programming Explained”, Addison Wesley, 2001.

[Boehm & Huang 03]
Boehm, B. and Huang, L.G., “Value-Based Software Engineering: A Case Study”, IEEE Computer, 2003, pp. 33-41.

[Boehm & Sullivan 03]
Boehm, B. and Sullivan, K (2003) “Value-based software engineering”, ICSE 2003 Tutorial, May 5.

[Carlshamre 02]

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

[Carlshamre et al. 01]
Carlshamre, P., Sandahk, K., Lindvall, M., Regnell, B. and Nattoch Dag, J., “An Industrial Survey of Requirements Interdependencies in Software Release Planning”. In: Proceedings of the 5th IEEE International Symposium on Requirements Engineering, 2001, pp. 84-91.

[Cockburn 02]
Cockburn, A., “Agile Software Development”, Pearson Education, 2002.

[Davis 03]
Davis, A.M., “The Art of Requirements Triage”. In: IEEE Computer 36(3), Mar 2003, pp 42- 49.

[DeGregorio 99]
DeGregorio, G., “Enterprise-wide Requirement & Decision Management”. Ninth Annual International Symposium of the International Council on System Engineering Brighton, England. June 6-10. 1999

[Favaro 02]
Favaro, J. (2002), “Managing Requirements for Business Value”, IEEE Software, March/April 2002, pp..22-24.

[Greer & Ruhe 03]
Greer, D and Ruhe, G., “Software Release Planning: An Evolutionary and Iterative Approach”. Vol. 46 (2004), pp 243-253.

[Gruenbacher 00]
Gruenbacher, P., “Collaborative Requirement Negotiation with Easy WinWin”. IEEE Software, 2000, pp. 954-958.

[Gurp, Bosch & Svahnberg 01]
Gurp, J. van. Bosch,J. and Svahnberg, M.. “On the Notion of Variability in Software Product Lines”. Proceedings of the IFIP Conference on Software Architecture, pp 45--54. IEEE Computer Society Press, 2001.

[Jung 98]
Jung, Ho-Won (1998) “Optimizing Value and Cost in Requirements Analysis”, IEEE Software, pp. 74-78.

[Karlsson et al. 98]
Karlsson, J., Wohlin, C and Regnell, B.,”An Evaluation of Methods for Prioritising Software Requirements”, Information and Software Technology 39 (1998), pp. 939-947.

[Karlsson et al. 03]
Karlsson, L., Regnell, B., Karlsson, J., Olsson, S., “Post-Release analysis of requirements selection quality – an industrial case study”,

[Natt et al. 01]
Natt och Dag, J., Regnell, B., Carlshamre, P., Andersson, M., Karlsson, J., “Evaluating automated for requirements similarity analysis in market-driven development”, 7th International Workshop on Requirements Engineering: Foundation for Software Quality, June 4-5 2001.

[Ngo-The & Ruhe 04]
Ngo-The, A. and Ruhe, G., “Maximally Diversified Solutions for Software Release Planning”, University of Calgary, Laboratory for Software Engineering Decision Support, Technical Report 10-2004.

[Nuseibeh & Easterbrook 00]

Nuseibeh, B., Easterbrook, S., “Requirements Engineering: A Roadmap”, in: The Future of Software Engineering, ICSE 2000, pp 35-46.

[Penny 02]
Penny, D., “An Estimation-Based Management Framework for Enhancive Maintenance in Commercial Software Products”. In: Proc. International Conference on Software Maintenance, pp. 122-130, 2002.

[Rittel & Webber 84]
Rittel, H., Webber, M., “Planning Problems are Wicked Problems. In: Cross, N. (ed.), Developments in Design Methodology. Wiley, Chichester, 1984, pp 135-144.

[Robertson et al. 03]
Robertson, J., Robertson, S., Orr, K., and Bennatan, E.M., “Next Practices in Requirements Engineering”, Cutter Consortium, 2003.

[Ruhe 03]
G. Ruhe: Software Engineering Decision Support - Methodology and Applications. In: ‘Innovations in Decision Support Systems’ (Ed. by Tonfoni and Jain), International Series on Advanced Intelligence, Volume 3, 2003, pp 143-174.

[Ruhe & Greer 03]
Ruhe, G., Greer, D., “Quantitative Studies in Software Release Planning under Risk and Resource Constraints”. In: Proceedings of the 2003 IEEE International Symposium on Empirical Software Engineering (ISESE 2003), pp 262 – 271.

[Ruhe & Ngo-The 04]
Ruhe, G., Ngo-The, A., “Hybrid Intelligence in Software Release Planning”. Journal of Hybrid Intelligent Systems, Vol. 1(2004), pp 99-110.

[Saaty 80]
Saaty T.L., The Analytic Hierarchy Process, Wiley, New York, 1980.

[Sharp, Finkelstein & Galal 99]
Sharp, H., Finkelstein, A., Galal, G.,”Stakeholder Identification in the Requirements Engineering Process”. Proceedings Tenth International Workshop on Database and Expert Systems Applications, 1999, pp. 387 –391.

[Standish Group Research 02]
Standish Group Research. “What are your Requirements?”, http://www.standishgroup.com/, 2002.

[Turner et. al 99]
Turner, C.R., Fuggetta, A., Lavazza, L., Wolf, A.L., “A Conceptual Basis for Feature Engineering”, The Journal of Systems and Software. 49, pp. 3-15, 1999.