

The Art and Science of Software Release Planning

Günther Ruhe and Moshood Omolade Saliu, University of Calgary

A hybrid release planning approach integrates the strength of computational intelligence and the knowledge and experience of human experts.

Incremental development provides customers with parts of a system early, so they receive both a sense of value and an opportunity to provide feedback early in the process. Each system release is thus a collection of features that the customer values. Furthermore, each release serves to fix defects detected in former product variants. Release planning (RP) addresses decisions related to selecting and assigning features to create a sequence of consecutive product releases that satisfies important technical, resource, budget, and risk constraints. A good release plan should

- provide maximum business value by offering the best possible blend of features in the right sequence of releases,
- satisfy the most important stakeholders involved,
- be feasible with available resources, and
- reflect existing dependencies between features.

Here we present two approaches to RP. The *art of RP* approach relies on human intuition, communication, and capabilities to negotiate between conflicting objectives and constraints. The *science of RP* approach formalizes the problem and applies computational algorithms to generate best solutions. We propose creating a synergy between the two, integrat-

ing human and computational intelligence to define optimal RP feature assignments.

The difficulties of creating a release plan

Developing a release plan once you've found an appropriate formal problem description is computationally complex, but even finding the description in the first place can be difficult owing to cognitive complexity. You need a proper understanding of the planning objectives and constraints as well as of the important stakeholders and their feature preferences. Although this information is often uncertain and changing, it doesn't mean RP is necessarily ad hoc and based on intuition.

Yet, if we perform a comparative analysis of existing methods, we see that most organizations select features informally.^{1,2} RP is typically done ad hoc—even when the planning involves

Not much is known about how to effectively and efficiently perform the planning process.

several hundred features, it's rarely based on sound models and methodologies. Furthermore, optimally selecting and scheduling features is inherently complex.^{3,4} Also, planning and follow-up replanning are time-consuming processes mainly owing to the need to elicit relevant information and negotiate compromises between stakeholders.^{5,6} Another problem is that the planning scope is often limited to just the next release.⁴

According to the Capability Maturity Model Integration, project management involves planning, monitoring, and controlling activities, and the planning process should establish and maintain plans that define project activities.⁷ This process includes developing the project plan, involving stakeholders appropriately, obtaining commitment to the plan, and maintaining the plan. Guidelines and standards (such as IEEE/EIA 12207.0) exist for the planning process in principle, but they don't explain how to operationally assign features to releases to ensure maximal business value. Not much is known about how to effectively and efficiently perform this process.

The art of release planning

This approach addresses RP's implicit and tacit aspects. To the extent that RP is an ill-defined problem, it requires human intuition and capabilities to clarify the problem before seeking a solution. One of the reasons for handling RP as an art can be an organization's lack of emphasis on processes in general, either intentionally or because they lack RP process maturity.

Agile development,⁸ which leverages the well-known advantages of small, iterative software releases to receive early customer feedback and avoid the "big bang" syndrome, also takes this more humanistic approach. RP in agile development focuses on planning for the next iteration, and this planning procedure relies on physical meetings between the important stakeholders to discuss and negotiate informally which features to develop next and how much effort they require. (Note that the number of features being considered is typically small.)

Unfortunately, RP in agile development doesn't provide guidance on how to decide on features and priorities when multiple stakeholders are involved.⁸ Nor does it suggest techniques to balance conflicting demands between multiple stakeholders.

Even in more plan-driven environments, RP is often done quite informally, relying on elementary tools such as spreadsheets. The release decisions are made by contacting the main stakeholders and manually balancing their interests and preferences with available resources. However, exponential growth in the number of possible release plans surpasses the power of manual plan generation, especially as the number of features and stakeholders grows.

The science of release planning

This approach is primarily based on the belief that we can (at least approximately) formalize the problem, and that solving this formalized problem will produce meaningful results. Some researchers have modeled the problem as a specialized optimization problem. In formulating an optimization model for RP, Anthony Bagnall and his colleagues assign weights to customers according to their importance to the software company.⁹ The objective is to find a subset of customers whose features must be satisfied (within the budget). Similarly, Ho-Won Jung's approach selects features that give maximum value for minimum cost, considering the software system's budgeted costs.¹⁰ These optimization approaches cope better with larger problem sizes, but they don't give customers an opportunity to participate in RP decisions, and they don't plan beyond a single release.

Attempts to formalize certain aspects of the RP problem while still using ad hoc methods include Mark Denne and Jane Cleland-Huang's *incremental funding method*¹¹ and David Penny's *maintenance planning*.¹² IFM differs from other RP approaches because of its focus on revenue projections, but it doesn't consider other RP constraints. Penny's approach focuses on release monitoring by ensuring a balance between required and available effort. However, neither approach formalizes RP's major impacting factors.

The formulation we present here extends one given elsewhere¹³ by offering greater flexibility in the number of releases it addresses. However, for simplicity, we only present the case of $K = 2$ releases.

Decision variables

Suppose we have a collection of features $F = \{f(1), f(2), \dots, f(n)\}$. This feature set might relate to new functionalities, customers' change

requests, or defect corrections. We generally refer to all three categories as a set. The goal is to assign the features to a finite number K of release options. This is described by decision variables $\{x(1), x(2), \dots, x(n)\}$, where $x(i) = k$ if we assign requirement i to release option $k \in \{1, 2, \dots, K\}$; otherwise, $x(i) = 0$.

Dependencies between features

Various dependencies between features are possible, and most features are involved in some sort of dependency relationship.¹⁴ In our formulation, we consider two types of dependency constraints: a coupling relation called C and a precedence relation called P. Coupled features should be released jointly because they depend on each other. Precedence features should be released in a certain order because offering a particular feature might not make sense if it depends on a related feature that won't appear until a later release.

Resource constraints

Resources are an essential part of RP. For feature realization, project managers typically must consider various resource constraints. Usually, these constraints relate to either budget or effort resources, and all constraints include bounds on the maximum capacities available for each resource type.

We assume that T resource types and capacities $\text{Cap}(k, t)$ relate to all releases $k = 1 \dots K$ and to all resource types $t = 1 \dots T$. Every feature $f(i)$ requires an amount $r(i, t)$ of resources of type t . Thus, each release plan x assigning feature $f(i)$ to release k expressed as $x(i) = k$ must satisfy

$$\sum_{x(i)=k} r(i, t) \leq \text{Cap}(k, t) \quad (1)$$

for all releases k and resource types t .

Stakeholders

Stakeholders are extremely important in product planning, but not all stakeholders have the same level of importance. Assume a set of stakeholder $S = \{S(1), \dots, S(q)\}$. We can assign a relative importance of $\lambda(p) \in \{1, \dots, 9\}$ to each stakeholder p . We use an ordinal nine-point scale to allow sufficient differentiation in the degree of importance, so $\lambda(p) \in \{1, 3, 5, 7, 9\}$ indicates very low, low, medium, high, and extremely high importance, respectively. Even numbers (such as $\lambda(p) = 2, 4, 6, 8$)

indicate a value between the preceding and succeeding odd-number values.

Feature prioritization

We can use different criteria to prioritize features. Our current formulation expresses priority in relation to value and urgency—again using the nine-point scale. For the value proposition, each stakeholder is asked to assign an ordinal value, $\text{value}(p, i) \in \{1, 2, \dots, 9\}$, to each feature based on its assumed (relative) impact on the product's overall value. So, $\text{value}(p, i) = 1$ and $\text{value}(p, i) = 9$ represent the lowest and highest values, respectively.

Time-to-market concerns regarding certain product features can motivate urgency. Assume we're planning two releases. Each stakeholder has nine votes for each feature, which we'll distribute among three possible options: *assign to release 1*, *assign to release 2*, or *postpone*. The higher the number of votes assigned to option k , the more satisfied the stakeholder will be if that feature is included in the respective option. So, an urgency vote $\text{urgency}(p, i) = (9, 0, 0)$ indicates that stakeholder $S(p)$ has assigned the highest possible urgency to feature $f(i)$ and thus wants to include it in the first release. A vote of $\text{urgency}(p, i) = (0, 9, 0)$ indicates a strong desire to include the feature in the second release, while $\text{urgency}(p, i) = (0, 0, 9)$ indicates a strong desire to postpone the feature. An urgency vote of $(3, 3, 3)$ expresses no urgency preference. Any combination of the three cases expresses different degrees of these three basic options.

Objective function

The planning objective is typically a mixture of different aspects such as value, urgency, risk, satisfaction or dissatisfaction, and return on investment. The explicit function's actual form (equation 2) tries to bring together the different aspects in a balanced way. In our model, we assume the following:

- An additive function exists in which the total objective function value is determined as the sum of the *weighted average satisfaction* WAS(i, k) of stakeholder priorities for all features $f(i)$ when assigned to release k .
- Each value WAS(i, k) is determined as the weighted average of the products of the two dimensions of prioritization described earlier.

Stakeholders are extremely important in product planning, but not all stakeholders have the same level of importance.

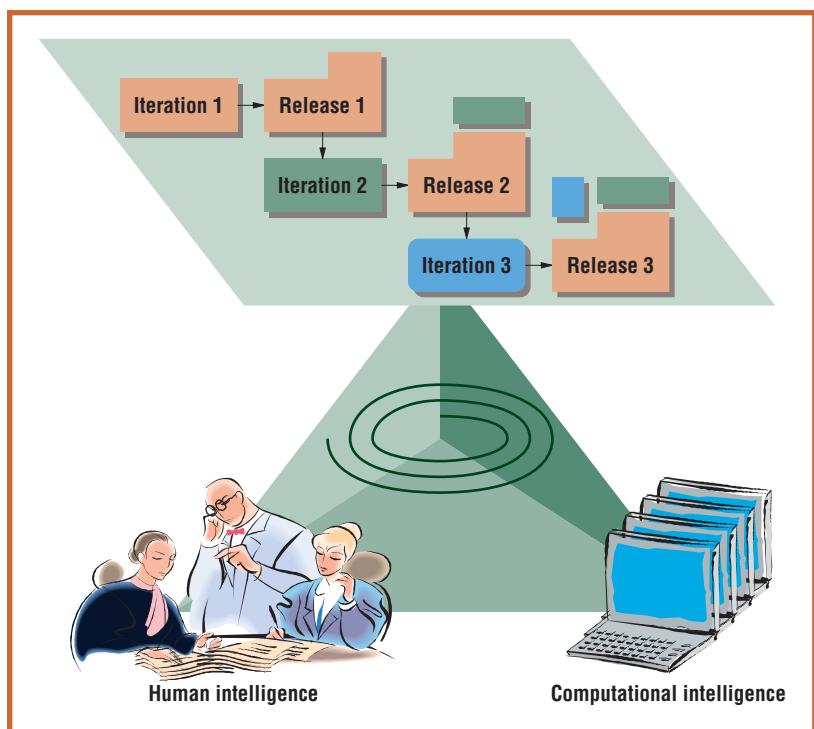


Figure 1. Our hybrid approach combines art and science in a high-level framework that stresses the continuous process needed to perform planning and replanning.

- Stakeholder $S(p)$'s degree of impact is determined by the relative importance $\lambda(p)$.
- For each release option k , normalized parameters $\xi(k)$ describe each option's relative importance.
- A vector of urgency preference for each stakeholder and each feature, given as $\text{urgency}(p, i) = (\text{urgency}(p, i, 1), \text{urgency}(p, i, 2), \text{urgency}(p, i, 3))$.

According to these assumptions, the objective is to maximize a function $F(x)$ among all release plans x subject to the satisfaction of resource constraints (described in equation 1) and dependency constraints (just given). $F(x)$ is given as

$$F(x) = \sum_{k=1 \dots K} \sum_{i: x(i)=k} \text{WAS}(i, k) \quad (2)$$

where

$$\text{WAS}(i, k) = \xi(k) [\sum_{p=1 \dots q} \lambda(p) \cdot \text{value}(p, i) \cdot \text{urgency}(p, i, k)] \quad (3)$$

A hybrid approach based on integer linear programming

The formalized description we present in equations 1 and 2 lets us apply efficient optimization algorithms that offer optimal or nearly optimal solutions. This fundamentally

differs from simple spreadsheet calculations, which support merely a series of elementary arithmetical operations. The problem is far too complex to achieve (and guarantee) quality release plans using just spreadsheets.

The given formulation of the RP problem constitutes a specialized integer linear programming problem.¹⁵ All the stated objectives and constraints are linear functions, and the decision variables are integers. The solution algorithms we've adopted are based on solving a sequence of linear programming problems without integer conditions. These relaxed problems are well understood and can be efficiently solved. We combine this approach with heuristics to achieve integrality and to generate not just one but a set of sufficiently good solutions. We consider a solution to be sufficiently good (or qualified) if it achieves at least 95 percent of the maximum objective function value. It's more practical to offer a small set of solutions, all of similar quality though different in their structure, than to determine just one plan for a vaguely defined problem.

So, our approach formulates a series of problems as variants of the original formal model. Then we solve these problem variants to generate a set of qualified alternative solutions. A human decision maker—such as the project manager—evaluates the solutions based on his or her experience and familiarity with the problem context.

In this way, art and science complement each other. The art-based approach has trouble coping with the RP problem's complexity as the number of factors grows. The science-based approach copes better with complexity but can't evaluate the problem with the same analytical abilities as the human decision maker. Our approach offers a high-level framework stressing the continuous process needed to perform planning and replanning (see figure 1). We further refine our approach by performing a variety of tasks during the three phases of the planning process—modeling, exploration, and consolidation.

Phase 1: Modeling

The top of the triangle in figure 1 addresses problem conceptualization. It focuses on the formal description of the dynamic real world to make it suitable for computational-intelligence-based solution techniques. This phase includes three main tasks.

Plan objectives and constraints. This task involves formalizing the objective function that models the underlying problem and all present constraints. We discussed the process of realizing this task when we discussed the instantiation of science-based RP modeling. All impacting factors and prioritization schemes are contained in the objective function and constraints.

Offer stakeholder voting. Stakeholders are the people who directly or indirectly influence or are influenced by the software release plans—the developers, managers, customers, and users. Giving them the opportunity to vote on features according to their preferences is important because they set the project’s course and decide the evaluation criteria for its success. This activity assigns a set of feature priorities according to stakeholder preferences.

Estimate resources. Estimates of the likely amounts of various resources needed to implement each feature must be determined during the modeling phase, because finding a solution to the objective function during exploration depends on it. Estimating the effort, cost, and time required has always been a major software engineering challenge. One reason for this is that estimates made during an early stage of the development life cycle are generally fraught with uncertainties. However, hybrid approaches that integrate the judgment of human experts with formal techniques have proven promising in this situation.

Phase 2: Exploration

During this phase, we generate the solution plan based on the formal model. Because of the model’s sophistication, we need efficient special-purpose solution techniques. We developed specialized integer programming algorithms to explore the solution space and generate solution alternatives.

Phase 3: Consolidation

During consolidation, the decision maker evaluates the computational algorithm’s solution alternatives based on experience and the problem context. This helps the decision maker better understand the problem. Then, if need be, he or she can modify parts of the underlying model or make some local decisions (perhaps preassigning some features to specific releases). Typically, these decisions reduce the

problem’s size and complexity for the next iteration. Several iterations are possible until a desirable solution alternative is derived.

After formalizing the problem, the decision maker can perform what-if scenarios and can base replanning on the results:

- What if some of the stakeholder weights change?
- What if some of the capacities are increased (or decreased)?
- What if we modify the objective function or some of the constraints?

A sample problem illustrating the proposed approach

Table 1 presents key project data for a sample project that illustrates our proposed approach. Our trial project encompasses 15 features to be included in the next two releases.

According to the project manager, the relative importance of releases 1 and 2 are $\xi(1) = 0.7$ and $\xi(2) = 0.3$, respectively. This indicates that release 1 is currently more important than release 2. For simplicity, only two stakeholders, $S(1)$ and $S(2)$, prioritize the features. Their degree of importance is $\lambda(1) = 4$ and $\lambda(2) = 6$, respectively.

Four resource types are involved in realizing the features. Each has total capacity bounds for the two release periods. We observe that the total demand of all features exceeds the total capacity, so we can’t implement all the features in the next two releases. Also, three coupling and five precedence constraints exist:

$$\begin{aligned} \{(7, 8), (9, 12), (13, 14)\} &\in C \\ \{(2, 1), (5, 6), (3, 11), (8, 9), (13, 15)\} &\in P \end{aligned}$$

For example, $(7, 8) \in C$ means we should implement features $f(7)$ and $f(8)$ in the same release, and $(2, 1) \in P$ means we should implement $f(2)$ before $f(1)$, although we could implement both in the same release.

Stakeholder urgency voting

To illustrate urgency voting, consider feature $f(1)$. Stakeholder $S(1)$ voted urgency($1, 1$) = $(5, 4, 0)$, implying that feature $f(1)$ is desired in release 1 but could also appear in release 2. On the same feature, $S(2)$ voted urgency($2, 1$) = $(0, 3, 6)$ —he or she would prefer to postpone the feature but would consider assigning it to release 2. This stakeholder is strongly against having this feature in release 1.

During consolidation, the decision maker evaluates the solution alternatives based on experience and the problem context.

Table I
Features, resource consumption, and stakeholder feature evaluations

Feature $f(i)$	Resources				Stakeholder $S(1)$		Stakeholder $S(2)$	
	Analyst & designers (hrs) $r(i,1)$	Developers (hrs) $r(i,2)$	QA (hrs) $r(i,3)$	Budget (US\$ in thousands) $r(i,4)$	Value value(1, i)	Urgency urgency(1, i)	Value value(2, i)	Urgency urgency(2, i)
1. Cost reduction of transceiver	150	120	20	1,000	6	(5, 4, 0)	2	(0, 3, 6)
2. Expand memory on BTS controller	75	10	8	200	7	(5, 0, 4)	5	(9, 0, 0)
3. FCC out-of-band emissions	400	100	20	200	9	(9, 0, 0)	3	(2, 7, 0)
4. Software quality initiative	450	100	40	0	5	(2, 7, 0)	7	(7, 2, 0)
5. USEast Inc., Feature 1	100	500	40	0	3	(7, 2, 0)	2	(9, 0, 0)
6. USEast Inc., Feature 2	200	400	25	25	9	(7, 2, 0)	3	(5, 4, 0)
7. China Feature 1	50	250	20	500	5	(9, 0, 0)	3	(2, 7, 0)
8. China Feature 2	60	120	19	200	7	(8, 1, 0)	1	(0, 0, 9)
9. 12-carrier BTS for China	280	150	40	1,500	6	(9, 0, 0)	5	(0, 8, 1)
10. Pole-mount packaging	200	300	40	500	2	(5, 4, 0)	1	(0, 0, 9)
11. Next-generation BTS	250	375	50	150	1	(8, 1, 0)	5	(0, 7, 2)
12. India BTS variant	100	300	25	50	3	(9, 0, 0)	7	(0, 6, 3)
13. Common feature 01	100	250	20	50	7	(9, 0, 0)	9	(9, 0, 0)
14. Common feature 02	0	100	15	0	8	(9, 0, 0)	3	(6, 3, 0)
15. Common feature 03	200	150	10	0	1	(0, 0, 9)	5	(3, 6, 0)
Total resource consumption	2,615	3,225	392	4,375				
Available capacity, Release 1	1,300	1,450	158	2,200				
Available capacity, Release 2	1,046	1,300	65	1,750				

Computing the objective function value

So how do we compute the objective function value for each release? The objective function value depends on the release to which each feature is assigned.

If we assign feature $f(1)$ to release 1, then

$$S(1): \text{value}(1, 1) = 6, \text{urgency}(1, 1, 1) = 5, \\ \lambda(1) = 4$$

$$S(2): \text{value}(2, 1) = 2, \text{urgency}(2, 1, 1) = 0, \\ \lambda(2) = 6$$

Using $\xi(1) = 0.7$, we can compute $\text{WAS}(1, 1)$ as $0.7[(4 \times 6 \times 5) + (6 \times 2 \times 0)] = 84.0$. On the other hand, if we assign $f(1)$ to release 2, then

$$S(1): \text{value}(1, 1) = 6, \text{urgency}(1, 1, 2) = 4, \\ \lambda(1) = 4$$

$$S(2): \text{value}(2, 1) = 2, \text{urgency}(2, 1, 2) = 3, \\ \lambda(2) = 6$$

Using $\xi(1) = 0.3$, we can compute $\text{WAS}(1, 2)$ as $0.3[(4 \times 6 \times 4) + (6 \times 2 \times 3)] = 39.6$. So, just looking at these two options locally, it seems more favorable to assign $f(1)$ to release 1.

For each release plan, the $F(x)$ sums the

WAS according to the assignment of all 15 features. The goal (as formulated in equations 2 and 3) is to maximize the result.

Assume we've determined the two qualified release plans shown in table 2. Both plans are within the 95 percent quality range with respect to the objective function. As an added value, the two are structurally different, giving the decision maker some additional flexibility.

Plans such as those generated in table 2 derive from the power of modeling the problem scientifically. The decision maker can then evaluate them based on the "art" knowledge and experience before making a final decision.

A marriage of art and science, such as the one we have proposed, promises to carry RP's state of the practice to a higher level, giving due consideration to

- recognizing the need for a more sophisticated methodology,
- introducing more formalism that lets organizations easily plan projects containing several hundred features,

- formulating RP objectives with several impacting criteria rather than concentrating only on the values of the features,
- letting human decision makers easily evaluate formally generated release plans so they don't have to deal with lots of information without much visibility into the problem's structure, and
- proactively evaluating possible planning strategies to better understand the impact of varying problem parameters.

We've implemented our decision-support approach as part of the system solution ReleasePlanner (www.releaseplanner.com). Our ongoing efforts are geared toward empirical studies to collect and analyze quantitative and qualitative measures for assessing our proposed technique's added value in industrial settings. We've initiated this research through two pilot industrial case studies,^{5,6} but a detailed discussion of these empirical study results would require a separate article. ☺

Acknowledgments

We thank the Alberta Informatics Circle of Research Excellence (iCORE) for its financial support of this research. We also thank the anonymous reviewers for their detailed and valuable comments.

References

1. J. Karlsson and K. Ryan, "Prioritizing Requirements Using a Cost-Value Approach," *IEEE Software*, vol. 14, no. 5, 1997, pp. 67-74.
2. L. Lehtola, M. Kauppinen, and S. Kujala, "Requirements Prioritization Challenges in Practice," *Proc. 5th Int'l Conf. Product-Focused Software Process Improvement (PROFES 04)*, LNCS 3009, Springer, 2004, pp. 497-508.
3. B. Regnell, P. Beremark, and O. Eklundh, "A Market-Driven Requirements Engineering Process—Results from an Industrial Process Improvement Programme," *Requirements Eng.*, vol. 3, no. 20, 1998, pp. 121-129.
4. P. Carlshamre, "Release Planning in Market-Driven Software Product Development: Provoking an Understanding," *Requirements Eng.*, vol. 7, no. 3, 2002, pp. 139-151.
5. A. Amandeep, G. Ruhe, and M. Stanford, "Intelligent Support for Software Release Planning," *Proc. 5th Int'l Conf. Product-Focused Software Process Improvement (PROFES 04)*, LNCS 3009, Springer, 2004, pp. 248-262.
6. J.A. Momoh, "Applying Intelligent Decision Support to Determine Operational Feasibility of Strategic Software Release Planning," master's thesis, Dept. of Electrical and Computer Eng., Univ. of Calgary, Canada, 2004.
7. CMMI Product Team, *Capability Maturity Model Integration (CMMI) Version 1.1 Staged Representation*, tech. report CMU/SEI-2002-TR-029, Carnegie Mellon Univ., 2002.
8. B.A. Nejmeh and I. Thomas, "Business-Driven Product Planning Using Feature Vectors and Increments," *IEEE Software*, vol. 9, no. 6, 2002, pp. 34-42.
9. A.J. Bagnall, V.J. Rayward-Smith, and J.M. Whittley, "The Next Release Problem," *Information and Software Technology*, vol. 43, no. 14, 2001, pp. 883-890.
10. H.-W. Jung, "Optimizing Value and Cost in Requirements Analysis," *IEEE Software*, vol. 15, no. 4, 1998, pp. 74-78.

Table 2

Two qualified release plan alternatives, listing the release to which each feature is assigned and each weighted average satisfaction

Feature $f(i)$	Release Plan x1		Release Plan x2	
	$x1(i)$	WAS(i,k)	$x2(i)$	WAS(i,k)
1. Cost reduction of transceiver	1	84.0	1	84.0
2. Expand memory on BTS controller	1	287.0	1	287.0
3. FCC out-of-band emissions	1	252.0	3	0.0
4. Software quality initiative	3	0.0	1	233.8
5. USEast, feature 1	1	134.4	3	0.0
6. USEast, feature 2	2	516.6	3	0.0
7. China feature 1	2	277.2	1	88.2
8. China feature 2	2	43.2	1	19.6
9. 12-carrier BTS for China	3	0.0	2	72.0
10. Pole-mount packaging	3	0.0	3	0.0
11. Next-generation BTS	3	0.0	3	0.0
12. India BTS variant	3	0.0	2	75.6
13. Common feature 01	1	37.8	1	516.6
14. Common feature 02	1	8.4	1	277.2
15. Common feature 03	2	54.0	2	54.0
Objective function value $F(x)$		1,694.6		1,708.0

11. M. Denne and J. Cleland-Huang, "The Incremental Funding Method: Data Driven Software Development," *IEEE Software*, vol. 21, no. 3, 2004, pp. 39-47.
12. D.A. Penny, "An Estimation-Based Management Framework for Enhancing Maintenance in Commercial Software Products," *Proc. Int'l Conf. Software Maintenance (ICSM 02)*, IEEE CS Press, 2002, pp. 122-130.
13. G. Ruhe and A. Ngo-The, "Hybrid Intelligence in Software Release Planning," *Int'l J. Hybrid Intelligent Systems*, vol. 1, no. 2, 2004, pp. 99-110.
14. P. Carlshamre et al., "An Industrial Survey of Requirements Interdependencies in Software Release Planning," *Proc. 5th IEEE Int'l Symp. Requirements Eng.*, IEEE CS Press, 2001, pp. 84-91.
15. L.A. Wolsey and G.L. Nemhauser, *Integer and Combinatorial Optimization*, John Wiley, 1998.

About the Authors



Günther Ruhe is the Industrial Research Chair in Software Engineering at the University of Calgary and is an iCORE (Informatics Circle of Research Excellence) professor. His research interests include software engineering decision support, software release planning, requirements and COTS selection, measurement, simulation, and empirical research. He's a member of the ACM, IEEE Computer Society, and German Computer Society GI. Contact him at ICT Bldg., Rm. 545, Lab for Software Eng. Decision Support, Univ. of Calgary, 2500 University Dr. NW, Calgary, Alberta, Canada T2N 1N4; ruhe@ucalgary.ca.

Moshood Omolade Saliu is a PhD candidate and an iCORE (Informatics Circle of Research Excellence) scholar in the Computer Science Department at the University of Calgary, Canada. His research interests include software metrics and measurement, software engineering decision support, software process-related issues, and soft computing. He received his MS in computer science from King Fahd University of Petroleum & Minerals, Saudi Arabia. He's a member of the IEEE Computer Society. Contact him at ICT Bldg., Rm. 633, Lab for Software Eng. Decision Support, Univ. of Calgary, 2500 University Dr. NW, Calgary, Alberta, Canada T2N 1N4; saliu@cpsc.ucalgary.ca.

