

Decision Support for Product Release Planning based on Robustness Analysis

Ahmed Al-Emran

Department of Electrical &
Computer Engineering, Schulich
School of Engineering
University of Calgary
Calgary, Canada
aalemran@ucalgary.ca

Dietmar Pfahl

Department of Informatics
University of Oslo
Oslo, Norway
dietmarp@ifi.uio.no

Guenther Ruhe

Department of Computer Science
and Department of Electrical &
Computer Engineering
University of Calgary
Calgary, Canada
ruhe@ucalgary.ca

Abstract — Decision-making in requirements engineering often needs to be done in the presence of uncertainty. Rigorous methods can help to increase the probability of making the most appropriate decisions under the given circumstances. Robustness is a measure for the degree of stability of a solution in case of changes in the problem parameters. This paper presents a method (called DECIDE_{Release}) that applies simulation-based analysis and multi-criteria decision analysis on top of the existing strategic release planning approach EVOLVE*. The purpose of DECIDE_{Release} is to qualify decision-making by pro-actively exploring the robustness of the operational plans of upcoming releases. Based on this analysis, the strategic release plan that is the most robust against assumed changes in planning parameters at operational level can be selected. As a proof-of-concept, the applicability of DECIDE_{Release} is demonstrated by an illustrative case study. Results from a survey administered to managers and engineers in industry indicate that the proposed method is perceived as useful in practice.

Keywords: Release planning, strategic planning, operational planning, decision support, robustness, simulation, multi-criteria decision analysis

I. INTRODUCTION

Requirements engineering is a decision-centric process [1]. Software release planning is the process of deciding which features should be selected for the next releases and – for the ones selected – which feature should be offered in which release. Release decisions address the early stage of a product’s life-cycle. Release decisions are part of product management, an emerging discipline that aims at improving the whole process of development ranging from product requirements to design, implementation, product launch, and service [2]. In its essence, release planning is the process of defining the functionality of a sequence of product releases as part of incremental development.

Release planning is conducted at two levels of granularity: (i) strategic level, and (ii) operational level. At the strategic level managers deal with prioritizing and assigning software features to successive releases such that both technical and resource constraints are met while maximizing stakeholder satisfaction. Once a strategic release plan is agreed upon, operational release planning (ORP) takes place focusing on the details of the implementation of the next release. ORP deals with the assignment of developers to tasks necessary to implement the features agreed upon for the upcoming release.

Decision problems in requirements engineering are uncertain by nature. This uncertainty refers to both the objects of study (i.e., features or requirements) as well as their characteristics (e.g., effort or importance) used for the purpose of planning. Uncertainty can arise from organizations, people, technologies, functionality, time, budget, and resources.

Robustness in decision-making refers to the stability of decisions in the presence of uncertainty [3]. A solution is considered to be *robust* if its selection remains valid also after changes in problem parameters have occurred. A key element of the proposed method DECIDE_{Release} is the assessment of the robustness of operational release plans. This information is then used for making strategic release planning decisions. The importance of robustness as part of finding optimized solutions for software engineering search problems was highlighted by Harman [4]. In the context of release planning, robustness of solutions implies stability of development processes and consistency in assigning developers to performing different tasks. The more robust a release plan, the less vulnerable it is with respect to estimation uncertainty in problem parameters.

To assess robustness, we systematically investigate the impact of uncertainty in estimates of effort (of different tasks of a feature) and productivity (of developers for different tasks). We then use this information for selecting a strategic release plan (among several alternatives) that is most robust against those changes. We capture estimation uncertainty by generating and analyzing probability distributions of properties associated with operational release plans.

In our proposed method DECIDE_{Release}, we start from a set of optimized and diversified strategic release plans. Then we aim at evaluating robustness of each candidate strategic release plan at operational level. We adopt an ORP risk analysis procedure and, as its engine, employ a Monte-Carlo enhanced process simulation model to systematically vary input parameters such as expected task effort and assumed developer productivity.

The implementation of DECIDE_{Release} builds upon three proven decision support tools: ReleasePlanner™ [5] for strategic release planning, DynaReP for conducting simulation-based analysis of operational implementability of strategic plans [6], and ELECTRE IS [7] for performing multi-criteria decision analysis.

As a proof-of-concept, the applicability of the proposed method is demonstrated by an illustrative case study. In

addition, results from a survey with participants from industry are reported.

The main contributions of the paper are:

- (i) Integrating strategic and operational release planning by a new method that facilitates strategic decision-making based on information extracted from pro-active analysis of the operational implementability of plans. The plans itself are obtained from a systematic, optimization-based planning method called EVOLVE* [8].
- (ii) Qualified support of decision-making through combination of proactive simulation-based analysis and application of multi-criteria decision analysis.
- (iii) Implementation and initial evaluation of the method.

The rest of the paper is structured as follows. Section II summarizes related work and provides motivation for the research presented. Section III provides the problem statement. Section IV describes the underlying methods of the overall approach and their relationships. Section V illustrates the applicability of the proposed method through a case study example. Section VI provides an initial evaluation of the method. Section VII discusses both the validity of evaluation results and limitations of the proposed method. Section VIII summarizes our research and presents directions of future research.

II. RELATED WORK AND MOTIVATION

The importance of decisions in requirements engineering (RE) has been discussed by Evans *et al.* [9]. The authors emphasize that it is important to recognize requirements as design decisions in order to achieve a fully integrated software system. Regnell *et al.* [10] further develop this idea with the claim “Requirements Mean Decisions!” and investigate issues and challenges for both descriptive and prescriptive research.

Based on the comprehensive analysis of research conducted in [11], Ngo-The and Ruhe proposed an agenda for future research in requirements engineering decisions support. One proposed direction is to advance engineering decision support methodology with emphasis on decisions under uncertainty. Robustness as studied from a decision-making perspective in [3] is one contribution to this direction.

Several approaches and strategies have been proposed to resolve issues related to requirement selection and prioritization in the area of software release planning. For a comparative analysis of 24 strategic release planning methods we refer to [12]. Only a few of the analyzed methods take operational planning aspects under consideration. Robustness has not been considered by any of the analyzed methods.

There exist two published approaches that resolve Strategic Release Planning (SRP) and ORP problems simultaneously. The first approach is an integer linear programming (ILP) based solution method proposed by Li *et al.* [13]. Another approach called OPTIMIZE_{RASORP} [14], combines a special structure ILP with genetic algorithms (GAs). As a result, it is offering operational planning with more detail than the approach proposed by Li *et al.* OPTIMIZE_{RASORP} considers tasks associated with features, a

pool of developers to carry out these tasks, the productivity of developers when performing these tasks, task overlapping and associated dependency and mappings between tasks and developers. Using these parameters the approach is capable of calculating operational plans for realizing sets of features with maximized release value.

Antoniol *et al.* [15] considered robustness in the context of software project planning by adding a queuing simulation model to the GA-based approach. The GA part aims at minimizing project completion time by determining the optimal (or near-optimal) order of work packages to be processed, allocating people to different maintenance teams and assigning teams to work packages. The queuing simulator analyzes the robustness of the project plan suggested by the GA part and feeds this information back to the GA part in order to find a more robust plan, if possible. This approach is applicable for software development and maintenance projects in general but there is no direct connection with SRP. Moreover, the approach considers only project duration and does not consider impact on other planning aspects like schedule instability and re-allocation of developers.

ProSim/ORP (Project Simulation for Operational Release Planning) [16] proposed by Al-Emran *et al.* is a simulation-based method to measure the impact of uncertainty on operational release plans. The emphasis is on the proposed method itself and the computation of the impact of uncertainty on release make-span (i.e., total release duration). In this paper, we use the same method as in [15] for our robustness analysis. However, only estimation related uncertainties (i.e., effort estimates for tasks and productivity estimates for developers) are considered here. On the other hand, we go beyond analyzing only make-span as an impacted factor by including two additional factors being resource re-allocation and task scheduling. More importantly, in this paper, we tie the results of robustness analysis on operational level to the decision-making on strategic level.

III. PROBLEM STATEMENT

A. Strategic Release Planning

The purpose of SRP is to determine which features will be implemented in which releases. In [8], SRP has been formulated as an optimization problem maximizing an objective function. The criteria for strategic release planning are project specific. Often, the objective(s) of planning are related to creating maximum business value, to minimizing time-to market of features, or to maximizing satisfaction of stakeholders with the proposed plans. In addition, there are several constraints related to resource capacities and to technological conditions which have to be satisfied by the plans.

B. Operational Implementability of Strategic Plans

The purpose of ORP is to assign resources to feature implementation tasks such that total release make-span to implement all pre-selected features is minimized under given process and project constraints. Re-planning on an

operational level involves re-allocation of resources to feature implementation tasks in the face of resource changes, feature changes, and observation of planning mistakes arising from inaccurate estimation. The study presented in this paper focuses on the operational level of release planning assuming that strategic release planning has been completed.

For each task, we need to estimate the *effort* required to complete tasks. The execution time of a task not only depends on the effort but also on the *productivity* of the developer assigned to a task. Thus, execution time can be different for the same task if carried out by developers with unequal productivities. Task execution time is calculated by dividing the effort for the task (i.e., estimated task effort) by the productivity of the developer assigned to that task.

The primary goal of make-span minimization of the ORP problem is implementation of all features in minimum total duration. Different tasks must be carried out in order to implement the features. For each of the feature dependent tasks, there is an associated effort estimate. There are task dependency constraints between tasks of the same feature. For the implementation of all tasks, a pool of developers is considered available. Each developer has estimated productivity rates per task type (such as design, coding or testing). For a more formal problem description we refer to [16].

C. Addressing Uncertainty by Diversification Robustness

Strategic release planning is considered a cognitively and computationally complex problem, as it is hard to be formulated and, once formulated, hard to be computationally solved (because of being NP-complete [17]). In this context, the diversification principle was suggested in [17] to offer a variety of (qualified) solutions generated from optimization-based planning approach EVOLVE*. The diversification principle as formulated in [18] states that:

A single solution of the cognitively complex problem RELEASE is less likely to reflect the real-world problem solving needs when compared to a portfolio of qualified solutions which are structurally diversified. This is not changed by the formal optimality of the single solution.

The quality of decisions made is assumed to be dependent of the quality of the input received. DECIDE_{Release} is a method aimed at generating additional information to make the final selection among a set of qualified and diversified release plans. This information comes from the pro-active analysis of the robustness of the operational implementability of the candidate plans. The utility of these qualified strategic plans is approximately the same. Complementing formal quality (expressed by the utility function value), robustness is considered to be a key aspect for decision-making. The assumption is that the more robust a plan, the more attractive it is.

D. Robustness

In order to decide which plan should be finally selected, the robustness of their operational implementability is considered. For considering robustness a specific strategic release plan x^* , possible changes in problem parameters are

taken into account. In our case, we consider changes in the effort estimates $effort(i,j)$ of the task $task(i,j)$ of feature $f(i)$ and changes in the estimated productivity $prod(k,j)$ of developers $dev(k)$ performing a task of type j . For this robustness analysis, we adopted the method ProSim/ORP described in [16]. A sequence of R simulated changes $\{Effort\}_{r=1..R}$ and $\{Productivity\}_{r=1..R}$ is assumed in order to compute the following robustness factors:

Definition 1: (Make-span robustness)

Make-span robustness is expressed by the distribution of R make-span sensitivity values and its corresponding mean and standard deviation. Make-span sensitivity captures the relative change in total duration which occurred due to the variation of parameter vectors *Effort* and *Productivity*. Higher make-span sensitivity implies a less robust solution in terms of its impact on make-span. A formal definition of make-span sensitivity is given in [19].

Definition 2: (Robustness in resource allocation)

Robustness in resource allocation is represented by the statistical distribution of the changes in the R developer-task assignment sensitivity values and the corresponding mean and standard deviation. Developer-task assignment sensitivity refers to the degree of re-assignment of developers due to variation in parameter vectors *Effort* and *Productivity*. The more often re-assignments occur the less robust is the solution in that respect. A formal definition of the degree of re-assignment of developers is given in [19].

Definition 3: (Schedule robustness)

Schedule robustness is expressed by the distribution of R schedule sensitivity values and the corresponding mean and standard deviation. Schedule sensitivity refers to the degree of change in a schedule due to the variation of parameter vectors *Effort* and *Productivity*. This involves computing differences in start-times and end-times of different feature development tasks. More schedule sensitivity implies a less schedule-robust solution. A formal definition of schedule sensitivity is given in [19].

E. The Selection Problem

We consider a set X^* of optimized strategic release plan alternatives. The alternatives are the results of applying the existing strategic planning approach EVOLVE*. For each of the plans x from X^* , we evaluate its operational implementation, i.e., the assignment of developers to a sequence of tasks such that the overall make-span for completing the next release is minimized.

We further assume a sequence of R changes $\{Effort\}_{r=1..R}$ and $\{Productivity\}_{r=1..R}$ in the *Effort* and *Productivity* parameters of the problem. The question studied is as follows:

From the given set of pre-selected product release plans, which of the plans is most robust against the assumed changes?

In this formulation, *robustness* is a composite of make-span, resource allocation and schedule robustness factors.

IV. THE SOLUTION APPROACH – DECIDE_{Release}

A. Overview

Figure 1 depicts the process steps of the proposed DECIDE_{Release} method that is structured in three main components. Starting with an initial set of features elicited from stakeholders by the product manager, a diversified and optimized solution set of optimal strategic release plans is generated by ReleasePlannerTM (which implements the original EVOLVE* method [8]). This solution set provides strategic planning options – one of which is to be selected.

The remaining two components perform additional analyses on top of the original EVOLVE* method extending it into DECIDE_{Release}. One of these components is DynaReP that checks the operational feasibility of each strategic planning option with respect to robustness. The other component, ELECTRE IS, assists in finding the best, i.e., most robust, operational solution. The robustness assessment of the operational solutions serves as feedback to the strategic release planning decisions for further improvement.

B. Strategic Release Planning

EVOLVE* [8] is an iterative solution method for SRP. Unlike other optimization methods for SRP in the literature, EVOLVE* determines a set of diversified solution alternatives instead of calculating just one (optimal) solution. Each of these alternatives has a proven level of quality in terms of the utility function.

The EVOLVE* method is implemented in the decision support system ReleasePlannerTM [5] which has been successfully applied in several industry projects. As its input, it requires information like the set of features with their estimated resource consumption, dependency among features, available resource capacity, number of releases to plan with their relative importance, stakeholders and their relative weights, prioritization of features by stakeholders, and precedence/coupling relationship among features. As its output, ReleasePlannerTM produces five optimized and diversified solution alternatives for strategic release planning.

C. Operational Release Planning and Robustness Analysis

Based on our previous work we solve the ORP problem by using a method called PRP (Planning/Re-Planning) and its integrated simulation model DynaReP (Dynamic Re-Planner) [6].

Solving the ORP problem means assigning developers to feature development tasks of the next release without violating process and project constraints. DynaReP is flexible enough to accommodate any proposed developer-to-task assignment heuristic. That is, one can plug-in any optimization method from the operational research literature that schedules tasks and assigns developers. Alternatively, one can also plug in assignment heuristics that follow an organization-specific development process and staffing policy. For example, in [16] DynaReP was adjusted to an organization-specific strategy for assigning developers to tasks.

Once operational planning is completed, we analyze the resulting plan (i.e., the *baseline* plan) with regards to robustness. Since operational planning inputs such as task efforts and developer productivities are susceptible to underestimation and overestimation, it is interesting to assess the robustness of the proposed operational release plan due to variation in these parameters.

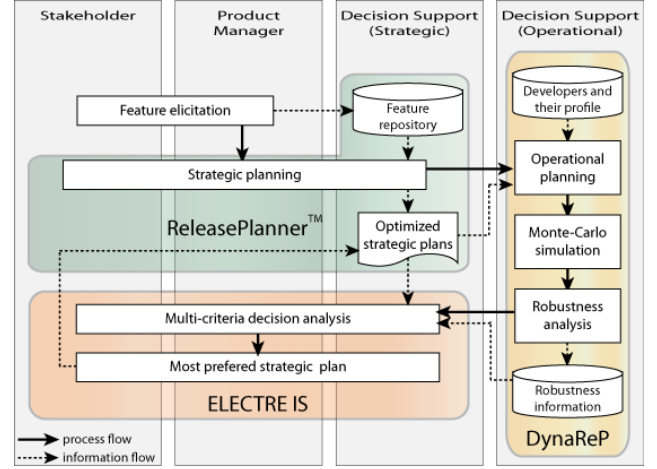


Figure 1. Process steps of DECIDE_{Release}

For the robustness analysis, we adopt the simulation-based risk analysis procedure ProSim/ORP (Project Simulation for Operational Release Planning) [16] and, being a simulation model, DynaReP can act as its core engine. DynaReP allows for systematic variation of planning parameters by defining probability distribution functions for each of the planning parameters. Values from these distribution functions can be sampled in each simulation run according to the Monte Carlo approach. A number of simulation runs is performed to quantify the respective impact on the operational release plans and results are represented as statistical profiles like average impacts and their standard deviations. Variations of task workload and developer productivity parameters result in variations of make-span (i.e. total duration of a release), allocation of developers to tasks (i.e., staffing), and structure (i.e., start/end times of tasks) of the baseline plan (i.e., the initial operational release plan before parameter variation is conducted). The resulting measurements of variation are recorded as (i) make-span robustness, (ii) resource re-allocation robustness, and (iii) schedule robustness, respectively.

D. Multi-criteria Decision Analysis

ELECTRE IS [20] is a method providing support to the problem of selecting one (or a set of) alternative(s) out of a given finite set of alternatives. It comprises two main procedures: (i) construction of one or several outranking relation(s) followed by (ii) an exploitation procedure.

We chose ELECTRE IS because it can handle uncertainty in preference building. When asked about what makes a good release plan, any project manager can come up with one or more evaluation criteria. However, especially

under the assumption of non-compensative criteria, it is difficult to combine these criteria into an objective function. ELECTRE IS supports customizing thresholds for what makes a difference between two alternatives, what means indifference, and what constitutes vague or no opinion.

The overall outranking relation is a relation $R(a,b)$ representing the degree to which each of the criteria supports the statement that “a is at least as good as b”. Each pair of alternatives (a, b) can be evaluated against a set of individual criteria. This idea is applied for a set of candidate strategic release plans. Each individual order relation (derived from a criterion) is assigned a weight representing the relative importance of the criterion in the overall comparison. The decision method then aggregates these relations using their weights to achieve a final outranking relation from which one (or a set of) alternative(s) can be identified as not outranked by any alternative. For a more formalized description we refer to [19].

E. Measurement of Robustness

In the following, we briefly describe how we compute the three robustness components from the outputs of R simulation runs.

Makespan Sensitivity captures the relative increase in make-span due to the variation in operational planning parameters like *Effort* and *Productivity*. *Effort* represents workload required to finish a feature development task (such as “flash implementation”, “database update”, and “java coding”) whereas *Productivity* indicates relative competence of a developer performing different tasks.

The formula to compute the magnitude of the make-span sensitivity for the r^{th} simulation run is:

$$\text{Makespan Sensitivity}_r = \frac{\text{makespan}_r - \text{makespan}_{\text{baseline}}}{\text{makespan}_{\text{baseline}}}$$

$\text{Makespan}_{\text{baseline}}$ is the make-span of an operational release plan using the initial estimated values (i.e., no variation) of parameters. Makespan_r is the make-span value of the operational release plan generated by the r^{th} simulation that samples values for *Effort* and *Productivity* parameters from a given probability distribution function. Make-span robustness is then expressed as the statistical distribution of R make-span sensitivity values and its corresponding mean and standard deviation.

Resource allocation robustness is expressed as the statistical distribution of R developer-task assignment sensitivity values and the corresponding mean and standard deviation. The developer-task assignment sensitivity of the r^{th} simulation is the number of re-assignments of developers to tasks. The number of re-assignments is computed by comparing the developer allocation computed by the r^{th} simulation with that of the baseline plan. The count of allocation differences is then divided by the maximum possible number of re-assignments.

Similarly, schedule robustness is computed as the distribution of R schedule sensitivity values and the corresponding mean and standard deviation. Schedule sensitivity is calculated based on the magnitude of changes

in start-times and end-times of feature development tasks between simulated plans and the baseline plan. Since the computation of schedule sensitivity is fairly complex we refer to [19] for details.

We would like to note that robustness analysis does not necessarily need to be restricted at the operational level. For example, we could vary the strategic input parameters (e.g., cost and value of each feature) based on a probability distribution function. The proposed method does not change principally if strategic robustness analysis would have been performed in addition to the operational robustness analysis.

V. ILLUSTRATIVE CASE STUDY

We now describe the overall $\text{DECIDE}_{\text{Release}}$ method through an illustrative case study example comprising a set of twenty candidate features and a set of six developers. First we perform the SRP analysis – using EVOLVE*. Afterwards we conduct ORP analyses including robustness analyses as well as multi-criteria decision analyses.

A. Strategic Analysis

1) Input Data

We use the same input data as for the benchmark problem studied in [8]. Table I shows the relative degrees of importance for each of the twenty candidate features $f(1)$ to $f(20)$ defined on a nine-point scale (nine being the highest priority) by four stakeholders called S(1) to S(4). The stakeholders are assigned weights 1, 3, 4, and 7, respectively. Estimated costs for implementing each feature are also specified. The considered cost constraints per release are 900 units.

Two releases, Release 1 and Release 2, are considered for strategic planning with relative importance 8 and 4, respectively. This means that delivering a feature in Release 1 contributes twice as much to the overall utility function value (see below for its principal structure) than delivering in Release 2. Table II lists technological constraints, i.e., dependencies between features. The dependency “ $f(i)$ precedes $f(j)$ ” implies that feature $f(i)$ cannot be assigned to a release later than the one feature $f(j)$ has been assigned to.

Planning at strategic level considers a set of N features $\{f(1), \dots, f(N)\}$ to be assigned to K releases. A release plan is characterized by a vector $x = (x(1), x(2), \dots, x(N))$ with $x(n) = k$ if, not violating any constraint, feature $f(n)$ is assigned to release k from $\{1, 2, \dots, K\}$; otherwise, $x(n) = 0$.

For this case study, we have used the following utility function (cf, [21] for a detailed description of utility functions for strategic release planning):

$TSFP(x) = \sum_{n=1 \dots N} SFP(n, x)$, given that

$SFP(n, x) = \sum_{k: x(n)=k} w_{re}(k) \cdot SCORE(n)$

$SCORE(n) = \sum_{p=1 \dots P} w_s(p) \cdot S(n, p)$, where

P is the number of stakeholders in the prioritization process

$w_r(k)$ is weighting factor for the releases k ($k = 1 \dots K$)

$w_s(p)$ is weighting factor for stakeholders p ($p = 1 \dots P$)

$S(n, p)$ is importance of stakeholders p for feature $f(n)$

Table I. Feature importance

f(n)	Feature Name	S(n,1)	S(n,2)	S(n,3)	S(n,4)	Cost
1	16 sector, 12 carrier BTS for China	5	4	8	7	100
2	China Feature 1	4	5	5	6	50
3	China Feature 2	1	3	1	3	60
4	China Feature 3	3	2	2	3	50
5	China Feature 4	2	4	1	6	60
6	China Feature 5	2	3	5	2	200
7	Common Feature 01	9	8	6	8	80
8	Common Feature 02	8	8	3	9	100
9	Common Feature 03	8	2	3	8	100
10	Common Feature 04	6	7	3	8	120
11	Cost Reduction of Transceiver	7	3	8	6	110
12	Expand Memory on BTS Controller	3	2	2	2	50
13	FCC Out-of-Band Emissions Regulatory Change	8	6	4	4	250
14	India BTS variant	3	3	3	2	200
15	India Market Entry Feature 1	3	6	4	8	70
16	India Market Entry Feature 2	8	5	1	3	80
17	India Market Entry Feature 3	2	2	6	4	60
18	Next Generation BTS 'In a Shoebox'	7	6	2	6	180
19	Pole Mount Packaging	7	3	4	5	200
20	Software Quality Initiative	9	9	9	9	120

Table II. Technological constraints

India Market Entry Feature	precedes	India Market Entry Feature 2
India Market Entry Feature	precedes	India Market Entry Feature 3
India BTS variant	precedes	Next Generation BTS 'In a Shoebox'
India BTS variant	precedes	Pole Mount Packaging

1) Results

With all the above information, five optimized and diversified SRP solutions are generated by ReleasePlannerTM as shown in Figure 2. Each solution maps features to subsequent releases. The entries “1” and “2” indicate

whether a feature has been assigned to the 1st or 2nd release. The entry “3” denotes that a feature could neither be accommodated in the 1st nor 2nd release, and hence has to be postponed. Figure 3 provides analytical information like degree of optimality and number of features implemented.

f(n)	Solution Set				
	Alternative 1	Alternative 2	Alternative 3	Alternative 4	Alternative 5
1	1	1	1	1	1
2	1	1	1	1	1
3	2	2	2	2	2
4	1	3	1	2	1
5	1	1	1	3	3
6	3	3	3	3	3
7	1	1	1	1	1
8	1	1	1	1	1
9	1	1	3	1	2
10	1	2	2	1	1
11	2	2	1	1	1
12	1	1	2	2	1
13	3	3	3	3	3
14	2	2	2	2	2
15	1	1	1	1	1
16	2	1	1	2	2
17	2	1	1	2	2
18	2	2	2	2	2
19	2	2	2	2	2
20	1	1	1	1	1

Figure 2. Five solution alternatives generated by ReleasePlannerTM






Criteria for Planning	Explanation	Alternative 1	Alternative 2	Alternative 3	Alternative 4	Alternative 5
9-Urgency	Degree of optimality					
	(Stakeholder feature points)	100.0% (5426)	97.2% (5275)	95.9% (5206)	95.6% (5189)	95.0% (5158)
Number of features assigned to...	Release	Alternative 1	Alternative 2	Alternative 3	Alternative 4	Alternative 5
	Release 1	11	11	11	9	10
	Release 2	7	6	6	8	7
	Total	18 of 20	17 of 20	17 of 20	17 of 20	17 of 20

Figure 3. Analytical information for strategic decision-making

B. Operational Analysis

1) Input Data

Each of the solution alternatives at strategic level is further analyzed at operational level from robustness perspective for both Release 1 and Release 2. Table III and Table IV show input information containing feature efforts and developer productivities, respectively, for each task. Once again, example tasks required to develop a feature could be “flash implementation”, “database update”, and “java coding”.

2) Results

Based on the information above, the simulation model DynaReP is applied to check for operational stability of each

of the five strategic solution alternatives in both releases. The results of this analysis are summarized in Table V. For this analysis, feature efforts shown in Table III and developer productivities shown in Table IV are considered as stochastic variables. 1000 simulation runs have been performed for each of which both factors are varied simultaneously by sampling from the following triangular distributions:

- Effort: (min, peak, max) \equiv (90%, 100%, 140%) and
- Productivity: (min, peak, max) \equiv (60%, 100%, 110%)

The resulting impacts on make-span, resource allocation, and schedule of tasks are reported as descriptive statistics indicating average values and related standard deviations. Each entry is computed based on the outcomes of 1000 simulation runs. For example, the make-span for Alternative

3 may increase by 47% on average as compared to the baseline plans of Releases 1 and 2 (i.e., without variation in workload and productivity). Similarly, the baseline schedules may change on average by 65.5% for Alternative 1 and 55.0% of task-developer allocation, on average, may need to be updated for Alternative 5. Therefore, the lower the values are, the better a solution alternative is, since lower values represent a more robust solution.

Table III. Estimated efforts for features development tasks

f(n)	Feature Name	Task1	Task2	Task3
1	16 sector, 12 carrier BTS for China	1	6	6
2	China Feature 1	8	4	2
3	China Feature 2	3	6	1
4	China Feature 3	6	2	3
5	China Feature 4	2	9	9
6	China Feature 5	5	6	4
7	Common Feature 01	1	10	8
8	Common Feature 02	1	4	6
9	Common Feature 03	5	4	4
10	Common Feature 04	7	5	1
11	Cost Reduction of Transceiver	10	5	6
12	Expand Memory on BTS Controller	5	1	7
13	FCC Out-of-Band Emissions Regulatory Change	4	9	9
14	India BTS variant	4	10	3
15	India Market Entry Feature 1	8	8	8
16	India Market Entry Feature 2	10	5	2
17	India Market Entry Feature 3	5	6	6
18	Next Generation BTS 'In a Shoebox'	10	10	6
19	Pole Mount Packaging	1	8	9
20	Software Quality Initiative	6	8	10

Table IV. Estimated productivity of developers for different task types

Developers	Task1	Task2	Task3
dev(1)	1.4	2	1.2
dev(2)	1	0	2
dev(3)	1	2	1
dev(4)	2	0	1
dev(5)	1	1.5	2
dev(6)	2	1	2

Table V. Simulative robustness analysis

Solution Alternatives	Robustness					
	Make-span		Resource Allocation		Schedule	
	AVG	SD	AVG	SD	AVG	SD
Alternative 1	46.5%	0.40	56.5%	0.12	65.5%	0.22
Alternative 2	45.5%	0.38	56.5%	0.13	57.5%	0.25
Alternative 3	47.0%	0.40	56.0%	0.14	58.5%	0.25
Alternative 4	53.0%	0.41	55.0%	0.13	65.5%	0.23
Alternative 5	49.0%	0.40	55.0%	0.13	65.0%	0.24

In order to build a preference structure among the five strategic alternatives, we apply multi-criteria decision analysis. To establish an overall outranking relation, we

assume that the following thresholds are applied to all criteria:

- A difference between two alternative values of less than 0.5 % is considered as indifference.
- A difference greater than 1.0 % is big enough to establish a difference
- A difference between 0.5 % and 1.0 % is vague and no classification can be made.

With this definition of preference, Figure 4 shows the resulting graph of outranking where all alternatives are represented by the vertices of the graph, and an arc is entered whenever one alternative is in favor compared to the other. From Figure 4, we observe that Alternative 1 outranks all other solutions. For more information on this computational procedure we refer to [19].

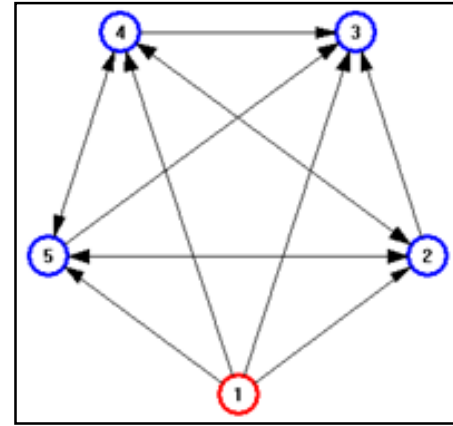


Figure 4. Graph of outranking

VI. EVALUATION OF DECIDE_{RELEASE}

A. Overview

In order to evaluate the overall DECIDE_{Release} method we have distributed a survey among twenty five people from industry as a starting point. All of them are either former graduates from the Software Engineering Decision Support Laboratory at the University of Calgary or past collaborators with this laboratory or people we know having sufficient understanding on the release planning and decision support topics.

The survey includes a compact version of the illustrative case study described in Section V to give the respondents of the survey an overall idea of the proposed method. It also asks respondents to answer multiple choice questions to elicit their opinion about value of DECIDE_{Release} (i.e., the extended method) compared to EVOLVE* (i.e., the original method).

B. Survey Design

In the survey (cf., [22] for a detail description), the set of all questions is divided into four parts – PART I to PART IV. PART I consists of the four background questions BQ1-BQ4 eliciting each respondent's educational background, experience level, interest and familiarity of the topic in general.

In PART II, we first present the strategic analysis part of DECIDE_{Release} which is same as the EVOLVE* method. Then we ask respondents to answer a general question GQ about the release decision and three applicability questions AQ1-AQ3 to assess confidence level, understanding, and comfortability in making decision. Thus, PART II is evaluating the EVOLVE* method.

In PART III we ask the respondents to answer the same set of questions as in PART II. However, this happens after presenting the operational analysis part i.e., robustness and multi-criteria decision analysis. This part of DECIDE_{Release} is the extension of the EVOLVE* method. We can evaluate this extension from the perception of the respondents by comparing answers of PART II and PART III.

At the end, in PART IV we ask the two effectiveness questions EQ1 and EQ2 to assess usefulness and added value of the overall DECIDE_{Release} method as perceived by the respondents.

C. Survey Results

We received in total thirteen responses out of twenty five requests to participate in the survey. Most of the respondents have an educational background in computer science or software engineering. The respondents' experience levels for requirements analysis, quality assurance, and project management varies from zero to fifteen years; for system design it varies from one to fifteen years; and for coding it varies from two to twenty-five years. Ten respondents replied that they have either high interest in project management or this is their major (i.e., primary) interest. None of the respondents possesses below medium familiarity with the software release planning process.

We analyzed all thirteen survey responses to compute the improvement achieved when using DECIDE_{Release} compared to using EVOLVE*. Our findings are as follows:

Finding 1: Impact on the release decision

Our presented result shows that six respondents changed their initial decision after robustness analysis information of DECIDE_{Release} had been provided. This clearly indicates that the method provided important enough information to the survey responders so that they gave a second thought about their decision and convinced half of them to change their minds. We consider this to be an important finding with regards to decision-making guidance.

In addition, the respondents' confidence level has improved on average. Figure 5 compares the improvement profile of both previous and proposed method. Each respondent assesses the confidence level about their decision in 9-point scale (9 being the highest confidence). On average, confidence in EVOLVE* is found to be 6.2 whereas confidence in DECIDE_{Release} is 7.0. Analyzing the 13 responses we can see that in six cases the respondents' levels of confidence in DECIDE_{Release} have increased compared to EVOLVE* while there is no case where confidence levels decreased.

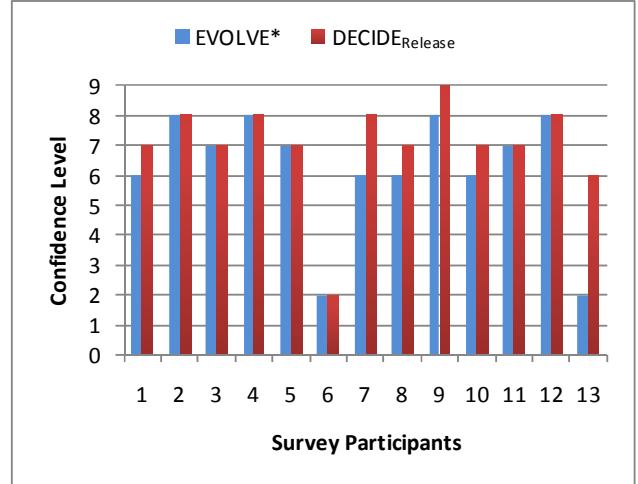


Figure 5. EVOLVE* vs. DECIDE_{Release} based on AQ1

Finding 2: Impact on the comfortability in making a decision

Six respondents found that making decisions with the help of DECIDE_{Release} is easier compared to EVOLVE* whereas four participants found it harder. Rest of the three participants did not find any difference.

Four participants found the results of DECIDE_{Release} are more understandable compared to only strategic information whereas five participants found that their understanding of produced results decreased. Since we also recoded the underlying reasons for making an evaluation decision, we found that those respondents whose understanding decreased the most did not actually understand the ORP process. Three other participants mentioned that they required more explanation of the kinds of information contained in the ORP related results produced by DECIDE_{Release}. However, at the end, there is no change in the average degree of understanding for the original (i.e., EVOLVE*) and the extended (i.e., DECIDE_{Release}) methods - both received 6.8 points on average by the respondents based on their degrees of understanding on produced results.

Finding 3: Impact on the satisfaction in making a decision

Ten respondents found DECIDE_{Release} useful and interesting and seven of them are willing to apply this extended method in their future release planning activities. The remaining three respondents answered that they were still in favor of their own release planning process; however, they thought that the robustness analysis contained in DECIDE_{Release} could be useful to increase confidence in their own method.

In total, eight respondents considered the robustness analysis to be an important aspect for release planning while the other five respondents mentioned that more information on how to apply the results of the robustness analysis is required to become useful for release planning.

VII. DISCUSSION

Decision-making in general and in particular for software requirements engineering is a complex task. Its results have a strong impact on success or failure of a project. Therefore it is essential to limit the shortcomings of intuitive decision-making [23]. The results from our case study and the related questionnaire about the different kinds of impact of the proposed method $\text{DECIDE}_{\text{Release}}$ provide a first indication that there is a good chance that it is able to support product managers and stakeholders in release planning decisions. There are, however, limitations with regards to the degree of validity of our approach and the results obtained from the survey conducted.

Construct validity: A central construct in our case study is the robustness of operational implementability of strategic plans. We capture robustness by varying two parameters (*Effort* and *Productivity*) and observing the effect of this variation on three responses (make-span, resource allocation, and schedule of the operational release plan). Since no generally accepted measures of robustness exist, we had to base the choice of our robustness properties on discussions with product managers of several companies (e.g., [24]).

Excessive variation in make-span, allocation of developers to tasks, and start and end times of tasks was uniformly considered as highly relevant by all decision makers in industrial organizations we talked to and who responded in the open part of the evaluation questionnaire. We account for this in our method by only reporting the degree of variation in terms of statistical properties of the distribution functions of response values and leaving the interpretation of this information to the decision-makers. We cannot claim that the type and number of factors and responses we have chosen are the best and only one choice. Other factors and response variables such as feature dependencies on operational level, communication and other forms of interaction between developers, cross-project dependencies of resource pools, to list just a few, might also be of importance but have not been considered in our case study.

Internal validity concerns the extent to which observed differences can be attributed to an experimental manipulation, and thus requires the consideration of competing explanations for an observed effect. Since our case study is heavily relying on a computerized simulation model, in principle, this should be the easiest type of validity to maximize. The simulated environment offers the experimenter a sterile setting in which entities adhere strictly to whatever heuristics they are assigned and within selected parameter bounds. Competing explanations should be completely controllable and the influence of confounding variables minimized.

In real-world experimentation external validity is the degree to which the findings in a local setting, containing a single set of sampling units, are applicable to the population of sampling units as well as other settings. In our particular case, external validity of our findings is limited in many ways. First of all, we used for most part of our case study a simulation model that is supposed to capture properties of

real-world developers as well as the mechanisms that are governing the decisions how developers are assigned to tasks and the ways how developers once assigned to a task perform this task. One way of increasing the external validity of our results would be to replicate the case study using other types of simulation models. Another way to increase the external validity of our findings would be to replicate the case study with other sets of features and associated properties (recall that we did our case study based exclusively on one – though real-world – set of 20 features). Even the number of 13 responses from running the questionnaire cannot be used to conclude external validity of the results, but as a first indication of validity which needs further evaluation.

Statistical conclusion validity pertains to the conclusions drawn from the statistical analyses, and the appropriateness of the statistical methods used in the analysis. Since we merely use descriptive statistics, statistical conclusion validity is not an important issue in our case study.

Empirical validity relates to the realism of the conclusions found in our case study and the related questionnaire. Though we are still lacking an in-depth investigation into the empirical validity of the simulation model used in our study, we can say that we have used the model in several real-world projects as a stand-alone tool in order to generate operational release plans. In all cases the results were shown to the responsible product managers. This was also confirmed by the results of the questionnaire.

The method demands quite some data to acquire. It is true that many organizations may not be able to provide these data since they do not measure and preserve them. However, there exist some mature organizations that possess the required data. For example, in [16] we worked with Chartwell Technology Inc. providing us with data for operational robustness analysis. In that case, acquisition of data for the case study was done with the assistance of two senior staff members at Chartwell with an average of 15 years of industrial experience. Both individuals were technical managers within the organization and thus intimately familiar with the technical details of the features included in the analyzed release in addition to being responsible for the management of those developers assigned to implement said features.

VIII. CONCLUSIONS AND FUTURE RESEARCH

Product release decisions are critical for the success of (incremental) software development projects. Providing the right set of features at the right time is one of the key factors for customer satisfaction. A decision support method for strategic product release planning is presented in this paper. The method extends the existing, optimization-based EVOLVE^* planning method by the additional application of simulation and multi-criteria robustness analysis. The new method supports product managers and stakeholders in selecting strategic release plans that are most robust to potential risks threatening the successful implementation of selected features.

As a proof-of-concept, we demonstrated the applicability of the proposed approach by a benchmark case study. We

show that the method is an improvement compared to the existing EVOLVE* as its decisions are now based on richer and more fine-grained information based on a systematic robustness analysis on the ORP level. This comes for the price of additional effort to conduct simulation based evaluation and to perform multi-criteria decision analysis.

Even though the idea of applying robustness analysis to task-developer scheduling problems is not new in the operational research community, application of this concept in the context of release planning in order to make strategic decisions has not been presented in the literature.

The advance approach is promising and was well received from the performed questionnaire conducted with representatives from industry. There are, however, several issues and limitations which are the content of future research. Firstly, the total effort of the method needs to be considered a limitation. Therefore, the method is primarily recommended to projects with a known high impact of improved decision-making where improvement is measured in term of both customer satisfaction (the objective of the EVOLVE* optimization) and robustness of results. Secondly, we will search for opportunities to apply the method in real-world software development settings. This will help us to get a better understanding of the way how we should represent robustness in our method and it will give us insights about the usability and relevance of the method for industrial product managers and stakeholders.

ACKNOWLEDGMENT

Part of the work presented was financially supported by the Informatics Circle of Research Excellence (iCORE) of Alberta in Canada, and the Natural Sciences and Engineering Research Council (NSERC) of Canada under Discovery Grant no. 327665-06 as well as Discovery Grant no. 250343-07. Ahmed Al-Emran would like to thank NSERC, iCORE, and the Killam Trust for their postgraduate/pre-doctoral scholarships. Many thanks to the survey participants for their valuable time and effort. All the comments provided by the anonymous reviewers help to make the paper better to understand.

REFERENCES

- [1] A. Aurum and C. Wohlin, "The Fundamental Nature of Requirement Engineering Activities as a Decision-Making Process," *Information and Software Technology*, vol. 45, Nov. 2003, pp. 945-954.
- [2] C. Ebert, "The Impacts of Software Product Management," *Journal of Systems and Software*, vol. 80, 2007, pp. 850-861.
- [3] B. Roy, "Robustness in Operational Research and Decision Aiding: A Multi-faceted Issue," *European Journal of Operational Research*, vol. 200, 2010, pp. 629-638.
- [4] M. Harman, "The Current State and Future of Search Based Software Engineering," *Proc. Future of Software Engineering, International Conference on Software Engineering*, 2007, pp. 342-357.
- [5] ReleasePlanner™, Expert Decisions Inc., www.releaseplanner.com (accessed: February 19, 2010).
- [6] A. Al-Emran, D. Pfahl, and G. Ruhe, "A Method for Re-Planning of Software Releases using Discrete-Event Simulation," *Software Process Improvement and Practice*, vol. 13, Jan./Feb. 2008, pp. 19-33.
- [7] ELECTRE IS, <http://www.lamsade.dauphine.fr> (accessed: March 16, 2009).
- [8] G. Ruhe, and A. Ngo-The, "Hybrid Intelligence in Software Release Planning," *International Journal of Hybrid Intelligent Systems*, vol. 1, Apr. 2004, pp. 99-110.
- [9] R. Evans, S. Park, and H. Alberts, "Decisions not Requirements: Decision-centered Engineering of Computer-based Systems," *Proc. of 1997 Workshop on Engineering of Computer-Based Systems*, Mar. 24-28, 1997, pp. 435-442.
- [10] B. Regnell, B. Paech, A. Aurum, C. Wohlin, A. Dutoit, and J. Natt och Dag, "Requirements Mean Decisions! – Research Issues for Understanding and Supporting Decision-making in Requirements Engineering," *Proc. of 1st Swedish Conference on Software Engineering Research and Practice*, Oct. 2001, pp. 49-52.
- [11] An Ngo-The, G. Ruhe, "Decision Support in Requirements Engineering, Engineering and Managing Software Requirements" (Ed. By A. Aurum and C. Wohlin), Springer 2005, pp 267-286.
- [12] M. Svahnberg, T. Gorchek, R. Feldt, R. Torkar, S.B. Selim, and M.U. Shafique, "A Systematic Review on Strategic Release Planning Models," *Information and Software Technology*, vol. 52, 2009, pp. 237-248.
- [13] C. Li, J.M. van den Akker, S. Brinkkemper, and G. Diepen, "Integrated Requirement Selection and Scheduling for the Release Planning of a Software Product," *Lecture Notes in Computer Science*, vol. 4542, 2007, pp. 93-108.
- [14] A. Ngo-The, and G. Ruhe, "Optimized Resource Allocation for Software Release Planning," *IEEE Transactions Software Engineering*, Vol. 35, 2009, pp. 109-123.
- [15] G. Antoniol, M. Di Penta, and M. Harman, "A Robust Search-based Approach to Project Management in the Presence of Abandonment, Rework, Error and Uncertainty," *Proc. 10th International Software Metrics Symposium*, IEEE Computer Society Press, Sep. 2004, pp. 172-183.
- [16] A. Al-Emran, P. Kapur, D. Pfahl, and G. Ruhe, "Studying the Impact of Uncertainty in Operational Release Planning – An Integrated Method and its Initial Evaluation," *Information and Software Technology*, vol. 52, 2010, pp. 446-461.
- [17] M.R. Garey, and D.S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," W H Freeman & Co, 1979.
- [18] G. Ruhe, A. Eberlein, and D. Pfahl, "Trade-off Analysis for Requirements Selection," *International Journal on Software Engineering and Knowledge Engineering*, vol. 13, 2003, pp. 345-366.
- [19] A. Al-Emran, D. Pfahl, and G. Ruhe, "A Hybrid Method for Advanced Decision Support in Strategic Product Release Planning," *Laboratory for Software Engineering Decision Support, SEDS-TR 088/2010*. Available at http://people.ualgary.ca/~aalemran/re2010/SEDS-TR_088_2010.pdf (accessed: February 19, 2010)
- [20] J. Figueira, V. Mousseau, and B. Roy, "ELECTRE methods." In *Multiple Criteria Decision Analysis: State of the Art Surveys*. Springer, New York, 2005, pp. 133-162.
- [21] G. Ruhe, *Product Release Planning: Methods, Tools and Applications*, CRC Press, 2010. ISBN: 9780849326202.
- [22] A Survey on the Impact of Robustness Analysis in Release Planning Decision-Making. Available at <http://people.ualgary.ca/~aalemran/re2010/Survey-RE10.pdf> (accessed: February 19, 2010)
- [23] L. Strigini, "Limiting the dangers of intuitive decision making," *IEEE Software*, vol. 13, Jan. 1996, pp. 101-103.
- [24] P. Kapur, A. Ngo- The, G. Ruhe, and A. Smith, "Optimized staffing for product releases and its application at Chartwell Technology," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 20, Sep. 2008, pp. 365-386.