A Prototype Tool Supporting When-to-release Decisions in Iterative Development

Jason Ho
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada
hott@ucalgary.ca

Shawn Shahnewaz
Department of ECE
University of Calgary
Calgary, Alberta, Canada
smshahne@ucalgary.ca

Guenther Ruhe
Department of Computer Science
Department of ECE
University of Calgary
Calgary, Alberta, Canada
ruhe@ucalgary.ca

***Abstract*— Shortening release cycles is one of the key elements for achieving highly competitive product releases. However, decisions about when-to-release are inherently complex: The potential competitive advantage through faster delivery needs to be balanced against the degree of readiness of the product (overall quality) and the added value through new and revised features. Pro-active analysis of the estimated impact of running through various release scenarios is expected to provide insights and essential inputs for the actual decision-making.**

When-to-release decisions are largely re-actively using existing release planning tools such as IBM Focal Point, On-time (for Scrum-based development) or ReleasePlanner. In this paper, the authors propose a plugin tool that analyzes the impact of varying the release date. More precisely, we proactively investigate the trade-off relationship between the total amount of implemented functionality and the predicted quality achieved from the related effort investment. As a result, the product managers are empowered to see the projected impact of releasing earlier (or later) in terms of reduced (or added) functionality and/or quality. As a proof-of-concept, we provide some preliminary results on the usage of the tool.

***Keywords*—Release engineering, when-to-release, prototype tool, decision support, software quality, software maintenance.**

I. Introduction

The lack of commitment from stakeholders and conflicting interests in release planning decisions was identified by Ebert and Brinkkemper [2] as one of the key reasons for delays in delivering software solutions. Deciding the proper release time is of key importance for the success of implementing and maintaining a software product [9]. The product manager often has to evaluate and decide among a large set of release alternatives. This is largely done by balancing between release duration, predicted quality of the release and the amount of functionality to be offered.

In this paper, we present a When-to-release Plugin (W2RP) that supports the pro-active analysis of a sequence of release scenarios defined by the user. In is able to provide support for answering questions such as:

- How many more (less) features will be implemented as the result of extending (reducing) the release -date?
- How to best compromise between creating new functionality and expected quality (measured in defects detected and fixed) of the release?

II. Background and Problem Statement

In this section, we provide the key concepts needed for the understanding of the tool's approach and workflow. For brevity, we refer to [4] and [10] for further details.

A. When-to-release Planning

A (product) feature is *a set of logically related requirements that enables the satisfaction of users' business objectives* [11]. In this paper, we examine feature at a high functionality level, instead of specific detailed requirements of implementation. When-to-release planning, therefore, is the problem of assigning a set of features to an upcoming release and deciding about the actual release date in consideration of the total release quality and release value. Both criteria -are defined below.

B. Total Release Value

The value of a feature is evaluated based on its ability to satisfy business objectives and users' requirements [11]. For our purposes, business value is determined on a nine- point scale based on the evaluation of an organization's stakeholders and product experts. This measure can easily be translated into projected revenue, prioritized features value, etc. which are often used in software industry [7].

The when-to-release decision making is based on the assumption that there is an existing plan comprising a set F_0 of features to be offered at the upcoming release with release date RD_0. This plan can be generated by existing release planning tools or being the result of manually planning.

In our model, the value, called *value(n)* of feature f(n) is defined as weighted average of priorities assigned to features and taken over all (weighted) criteria and for all (weighted) stakeholders [10]. The *Total Release Value* $TRV(F_0)$ is defined as the sum of all values of individual features.

$$TRV(F_0) = \sum_{f(n) \text{ from } F_0} value(n) \quad (1)$$

In the ReleasePlanner tool, cross-cutting, integrated features are considered, as long as they are designed and provided together with accurate effort estimates.

C. Total Quality of a Release

In this paper, the notion of software quality is associated with the number of defects found and fixed before the next release. Similarly, we approximate expected quality of a release through the result of the effort invested in testing. This concept is extensively studied and utilized under software reliability growth models [12]. Current research trends show that test effort dependent reliability growth models are effective and meaningful to the industry practitioners [2]. Test effort based quality models are mainly grouped into two classes of models: concave and S-shaped, as illustrated in Fig. 1. Both models fundamentally indicate that the defect detection rate decreases as the number of defect detected (and fixed) increases, and the total number of defects detected approach a finite value -close to zero. Details about these models can be found in [12].

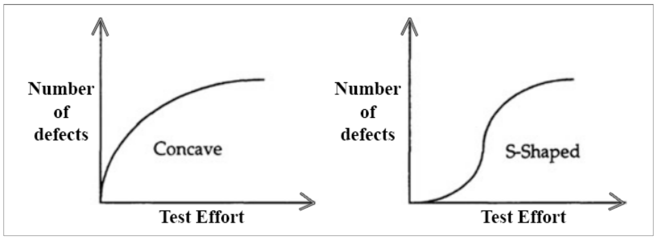


Figure 1: Quality model per feature based on test efforts

For each feature in all the possible pool of features F, based on its initial complexity estimate, a specific test effort is allocated. The expected release quality of the individual feature can be determined by selecting an appropriate quality model, either concave or S-shaped model.

Let Q(n, a_n, b, t_n) denote *the estimated quality of feature f(n)* based on the selected model. Therein,

- t_n is the estimated test effort for the feature f(n),
- a_n is expected number of defects for f(n) and
- b is the context specific shape factor of the selected model. By varying b, the shape of S – its concavity (Figure 1) can be adjusted for a project.

The *total release quality* can be defined as the aggregation of the individual feature quality models:

$$TRQ(F_0) = \text{Aggregation } \{Q(a_i, b, t\_effort_i): i = 1 \ldots n\} \quad (2)$$

Equation (2) implies that, by varying the test effort from the features, we can estimate the minimum and maximum release quality by aggregating the quality values of each features (based on their respective expected defects found and fixed). Therein, the selection of the appropriate aggregation operator is context specific. In the context of information fusion, many aggregation operators are proposed such as *Arithmetic Average*, *Geometric Average*, *Weighted Average*, and *Ordered Weighted average* [13].

In this work, we employed the geometric average for aggregating the individual quality of the features. As the ranges of the individual feature quality differ, geometric average provides a meaningful aggregated expected release quality. The model can be easily adapted to other aggregation operators, depending on project context.

D. Problem Statement

Based on the initial (*baseline*) when-to-release plan (RD_0, F_0) characterized by a release date RD_0 and a feature set F_0, we study a series of release scenarios from varying the release date. Each of these scenarios is a variation of the original release plan, and is determined from (RD_0, F_0) by re-balancing the effort allocated to testing (impacting expected quality) versus functionality implementable within the release duration RD_i.

As described in the pseudo-code (Table 1), we follow a greedy approach. Efforts from testing or implementation are exchanged, up to the threshold of (pre-set) reduction factors, and new solutions are generated. The *when-to-release problem* [4] is to determine as set of trade-off release plans generated from a series scenarios defined by the user from this exchange.

III. Related Work

Software reliability guarantees that software will work without failure for a specific time [3]. As software systems get more complex, completely removing all defects is challenging. The underlying assumption is that during the testing phase, correction of errors or bugs does not introduce any new errors and reliability of the overall software increases as bugs are discovered and fixed [9]. However, if testing takes too much time, the product may go over budget and miss the window of (business) opportunity [7] [8].

Currently, there is a lack of software tools that address the when-to-release problem as described here. McElroy & Ruhe [6] studied when-to-release decisions by allowing time-dependent value functions and adjusted resource capacities to determine value-risk tradeoff solutions. Ho & Ruhe [4] have proposed an approach for when-to-release trade-off, which is a predecessor to this method, utilizing a simplified quality-effort quantification formula.

Existing tools such as OnTime (www.ontimenow.com) focus on (Scrum) project management and tasks monitoring, with timeline being planned manually by the manager, based on pre-determined release duration, without the ability to provide alternative (trade-off) solutions between plans. The W2RP addresses precisely this issue, enabling informed decision-making process in release planning, with predictable and manageable impacts on release time, business value, and quality.

IV. W2RP Methodology and Implementation

A. Proposed Workflow

When-to-release planning follows an explorative method. Three principal use cases are possible, with ΔT is the maximum number of (work) days the release duration can be changed, $\Delta V(F_i)$ is the change in value due to (number of) functionality change, and $\Delta Q(F_i)$ is the change in quality due to the change in test cases (during i-th scenario). Users can configure the feasible range of changes for all three parameters, based on historical data or project specific needs. These factors will control the degree of acceptable change in release time, value and quality.

W2RP's workflow is shown in Figure 2 and further illustrated in Section V. The scenario playing environment assumes a baseline plan described by a release date RD_0 and a corresponding set of features F_0. This plan can be created manually or being the result of the application of any of the existing tools available to generate release plans [10].

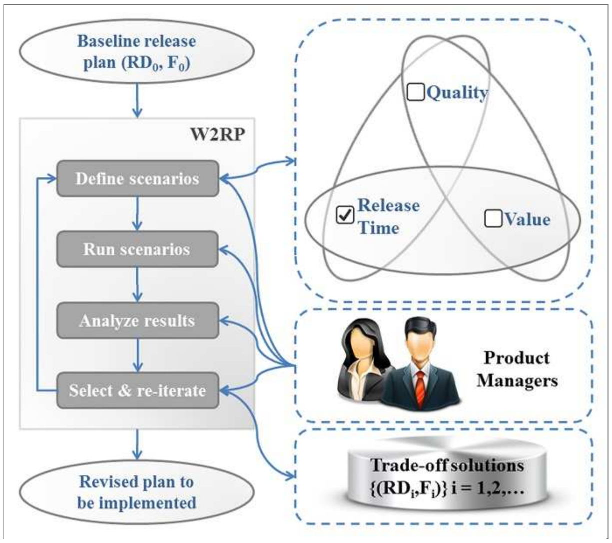


Figure 2: W2RP Workflow for fixed release date

Define scenarios allows specification of relative changes to be made against the baseline plan. In each of the use case scenarios, one of the parameters from the set of *{release time, release quality, release value}* is fixed to a variation of the respective baseline values. The new value (e. g. shorter release duration), is taken from the specified interval of exploration. In *Run scenario*, the user can interactively view the implications of varying among the remaining criteria, e.g. re-allocating testing effort or modifying the set of features to be offered.

From all the scenarios explored, a pool of release plans is generated. *Analyze results* eliminates all plans that are dominated by another plan. All trade-off solutions will be maintained in a pool of candidate release planning alternatives. In *Select and re-iterate*, the product manager(s) can either go back to define another scenario or terminate the scenario playing process and select the most attractive plan, representing the best balance between the competing criteria.

B. Prototype Implementation

As a proof-of-concept, we utilized the implementation of the above W2RP process as a prototype plugin on an existing release planning tool called ReleasePlanner [10]. ReleasePlanner was chosen because of its capabilities in term of analytical release planning. Based on a set of initial pool of features F, the tool can gather and prioritize releases based on (often conflicting) voice of customers. Furthermore, product managers can view, save, and compare different release plans, based on the resource allocation, features constraints, and view the predicted optimality of the release (based on customers' satisfaction). We then utilized and evaluated the plugin, coupled with the increase in the efficiency of the existing tool in a real life software project.

Table 1: Pseudo code of the W2RP core algorithm

Function Generate Solutions (F_0:=Baseline plan, F:= Features pool, α:= Quality reduction factor , ΔT:=change in release duration, RD: Release Date)

Define: S := Solution pool, S* := Trade-off solution

Step 1: Generating solution pool S

Case 1: $\Delta T = RD' - RD_0 < 0$

- While $|\Delta T| > 0$, $|TRQ'-TRQ_0| < \alpha$ do
- Calculate F' by:
  - reduce the low valued feature(s) from F_0 *OR*
  - reducing test effort from low valued feature(s)
- Add F' to S

Case 2: $\Delta T = RD' - RD_0 \geq 0$

- While $\Delta T \geq 0$, $|TRQ'-TRQ_0|<\alpha$ do
- Calculate new features set F' by:
  - increase test effort of high valued feature(s) of F_0 *OR*
  - add high valued feature(s) from features pool F.
- Add F' to solution pool S

Step2: Generating trade off solutions S*

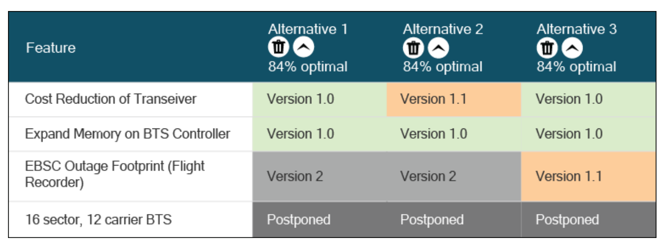
- Compare pair-wise every solution in S
- If there is no better solution than F_i, add F_i to S*

Step 3: Return trade-off solutions S*

C. Evaluation

We evaluated the implementation (in Table 1) using data from a sample project, which is a Scrum-type development project consisting of 22 features in the initial release F_0, selected as a baseline [4]. The plan was previously generated from ReleasePlanner. In consideration of all the resources needed and the capacities available, the release duration RD_0 was initially determined to be 80 days.

The product manager would like to predict the implications of varying release time, in this case, earlier up to ΔT = 15 days. The step-by-step execution of this example is included in a video guide walkthrough [14].



Feature	Alternative 1 84% optimal	Alternative 2 84% optimal	Alternative 3 84% optimal
Cost Reduction of Transeiver	Version 1.0	Version 1.1	Version 1.0
Expand Memory on BTS Controller	Version 1.0	Version 1.0	Version 1.0
EBSC Outage Footprint (Flight Recorder)	Version 2	Version 2	Version 1.1
16 sector, 12 carrier BTS	Postponed	Postponed	Postponed

Figure *3*: Trade-off solutions in comparison mode

W2RP allows managers to interactively choose the features set they want to be implemented (from the initial feature set F_0), specify a new release date, or update the effort allocation to each major activity of the release implementation process. The tool then generates a set of possible plans (according to the described algorithm), coupled with their degree of feasibility and optimality, and built on the existing tool, in this case ReleasePlanner. This is shown in Figure 3, where each plan is laid out with each feature being able to interactively assign to release plans, coupled with optimality.

Visually and interactively, the tool provides a bubble chart for easy comparison and selection between alternative plans (see Figure 4). The chart suggests the expected quality of each plan (very high, high, and good quality) as compared to the baseline plan. Each plan has a 3-point coordinate of: change in release duration (work days), total release value (point values), and expected quality standards. In this case, the highlighted plan (red circle) is selected by product manager since it provides good value, very high expected quality and faster release by 9 days.

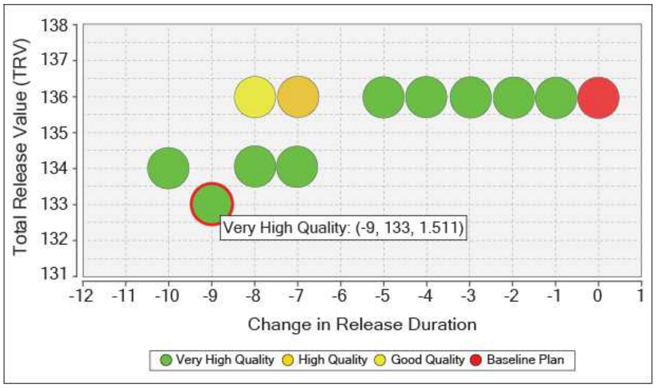


Figure 4: Visualization of trade-off solutions

D. Strengths and Limitations

The prototype plugin tool W2RP serves as a proof-of-concept for the potential of shortening release durations with predictable and manageable change in functionality and quality. As there is no existing tool available having similar capabilities, it was not possible to compare the tool with another one at the moment.

W2RP is effective in generating alternative scenarios. From varying project parameters (e.g. estimated defects, size and complexity of features, testing effort estimates, etc.), the tool allows analyzing sequences of scenarios. As the tool can be integrated into existing planning tools, not just limited to ReleasePlanner, giving it the ability to improve existing release plans, and gather valuable data.

One key threat to construct validity of this approach is the modeling and measurement of quality and value. Quality modeling and qualification is complex, especially in the case of integrated testing scenarios of large systems. The co-relation between testing effort and quality, features and business values need to be further examined and quantified. The application of the tool requires extensive data about effort, defect detections and testing efficiency. The better and the more reliable the project data, the better the planning result. However, even for uncertain data, using the tool for some pro-active analysis is helpful and can also be done under varying modeling assumptions.

V. Conclusions and Future Work

This paper outlined an approach, and implementation of the when-to-release tool W2RP. The decision support tool allows to pro-actively investigating a user defined sequence of release scenarios. As the result, product managers will be presented with a set of alternatives about the implications of varying the originally proposed release date. The impact will be visible as an update on the predicted total release value or by an update on the predicted total release quality.

In this paper, values and quality are modeled based on estimated efforts. Other quality constraints and business requirements, such as that of performance testing, and technical debts, are also considered in the potential area of work and application in future works as part of the modeling.

The prototype tool needs further analysis and evaluation of its acceptance, applicability and usefulness. Currently, the tool is integrated to existing industrial bug tracking tools such as JIRA and release planning tools to collect and analyze quality and requirements data from these tools in real time. The W2RP concept can also be used as a frequent job to re-optimize the solution set as new data becomes available, in real-time. Future work is needed to better estimate quality and to make the underlying development process dynamic by allowing changing parameters.

References

[1] B. Boehm, V. R. Basili, "Defect Reduction Top 10 List", in *Computer,* vol. 34, no. 1, 2001, pp. 135-137.

[2] C. Ebert, S. Brinkkemper, "Software product management – An industry evaluation" in *The Journal of Systems and Software,* 2014 , http://dx.doi.org/10.1016/j.jss.2013.12.042

[3] B.H. Far, "Software Reliability Models", in *Dependability & Reliability of Software Systems* (LN U of Calgary), 2012.

[4] J. Ho, G. Ruhe, "Releasing Sooner or Later: An Optimization Approach and Its Case Study Evaluation", in *Proceedings Workshop RELENG on Release Engineering at ICSE*, 2013.

[5] R. Lai, G. Mohit, P. K. Kapur, "A Study of When to Release a Software Product from the Perspective of Software Reliability Models" in *Journal of Software,* vol. 6, 2011, pp. 651-661.

[6] J. McElroy, G. Ruhe, "When-to-release decisions for features with time-dependent value functions", in *Requirements Engineering Journal,* vol. 15, 2010, pp. 337-358.

[7] C. Morris A., J. Eliasberg, T.H. Ho, "New product development: The performance and time-to-market tradeoff." in *Management Science* vol. 42.2, 1996, pp. 173-186.

[8] H. Ohtera, S. Yamada, "Optimum Software-Release Time Considering an Error-Detection Phenomenon During Operation," in *IEEE Trans. Reliability*, vol. 39, 1990.

[9] R. Peng, et al. "Testing effort dependent software reliability model for imperfect debugging process considering both detection and correction" in *Reliability Engineering & System Safety*, 2014.

[10] G. Ruhe, "Product Release Planning: Methods, Tools and Applications", CRC Press, 2010.

[11] K.E. Wiegers, "Software requirements," Microsoft Press, 2009.

[12] A. Wood, "Software reliability growth models: assumptions vs. reality" in *Proceedings from the Eighth International Symposium on Software Reliability Engineering. IEEE.* 1997.

[13] Z.S. Xu, Q.L. Da, "An overview of operators for aggregating information" in *International Journal of Intelligent Systems*, vol 18, 2003, pp. 953–969.

[14] https://sites.google.com/site/trongtanho/research/w2rp, Research case study data, last accessed Feb 2014.