Industrial Evaluation of the Impact of Quality-Driven Release Planning

Michael Felderer
University of Innsbruck
Innsbruck, Austria
michael.felderer@uibk.ac.at

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

Armin Beer
BVA & Beer Test Consulting
Vienna, Austria
armin.beer@bva.at

Guenther Ruhe
University of Calgary
Calgary, Alberta, Canada
ruhe@ucalgary.ca

ABSTRACT

Context: Product release decisions are often made ad hoc or not relying on up-to-date information from systematic and analysis-driven process. Often, too much emphasis solely is put into functionality, thereby neglecting the different quality aspects being important for the success of the product. **Objective:** For a case study project, we quantitatively measure the impact of the analytical and systematic release planning approach Q-EVOLVE II. **Method:** As an explorative and retrospective case study, we perform optimized planning scenarios and compare them with the actual baseline scenario. **Results:** For the case study project in an Austrian public health insurance institution, we demonstrate that the analytical and systematic approach Q-EVOLVE II results in a (1) reduced release time, (2) more features implemented in the same time, and (3) better final quality through increased testing activities and less defects slipping through at the time of release. **Conclusion:** In the context of the studied institution, analytical and systematic planning of product releases considering quality criteria is superior to ad hoc planning ignoring quality.

Categories and Subject Descriptors

K.6.3 [**Software Management**]: Software process. K.6.4 [**System Management**]: Quality assurance.

General Terms

Management, Measurement, Experimentation, Verification.

Keywords

Release Engineering, Release Planning, Test Management, System Testing, Software Quality, Decision Support, Case Study.

1. INTRODUCTION

Product release decisions (what to release? when to release? how good is the release?) are inherently complex because of their comprehensive information needs, the diversity of criteria, the variety of stakeholders involvement, and the presence of all types of resource and dependency constraints that have to be taken into account. In practice, these decisions are often made ad hoc [1] or are not relying on up-to-date data and a well-defined process [2]. Often, too much emphasis solely is put into functionality, thereby neglecting the different quality aspects being important for the success of the product. In this paper, we improve quality by discovering and eliminating defects from performing systematic testing. Lack of testing activities typically results in a large number of defects slipping through to the customer. This reduces the value of the delivered product and significantly increases the cost needed for bug fixing [3]. Testing, or system testing, in this respect is the process of planning, designing, and executing test cases to dynamically validate whether the system fulfills its specification [4].

Real-world examples of insufficient product management have been described by Ebert and Brinkkemper [5]. In the process of performing the release planning of software products for an Austrian public health insurance institution, we regularly observed that implementation of feature requests and change requests was prioritized over bug fixing, resulting in massive quality problems. Although there is a common agreement in the institution that sufficient testing of features has a positive impact on the released product, quality aspects are nevertheless often neglected in planning. Statements from the project and test management indicate that the main reason for this is due to late and unsystematic quality investments. To motivate and enable these investments, it is necessary to consider decision support (e.g. in meetings of the change control board), that takes testing and its impact on product quality from the very beginning of release planning.

In this paper, we report about a case study for applying a methodology to perform quality-driven release planning. It aims at balancing between development and test resources. Based on the analysis of effort, defect and features data, we performed a retrospective analysis and show how early investments into quality can improve later results. Improvements were demonstrated in terms of (1) reduced release time, (2) more features implemented in the same time, and (3) better final quality (less defects slipping through).

Section 2 describes related work. Section 3 presents the background for our two-staged quality-driven release planning approach and the related research questions. The design of the case study is the content of Section 4, with the results being presented in Section 5. The applicability of the results is discussed in Section 6. Finally, Section 7 concludes and gives directions for future work.

ESEM'14, September 18–19, 2014, Torino, Italy.
Copyright 2014 ACM 978-1-4503-2774-9/14/09…$15.00.
http://dx.doi.org/10.1145/2652524.2652579

2. RELATED WORK

Release planning addresses decisions related to the selection and assignment of features or requirements to a sequence of consecutive product releases such that the most important constraints are met [1]. The quality of features is validated by acceptance testing with respect to requirements to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system [6].

Recently, the alignment of requirements engineering and testing gained great interest in empirical research and practice [7][8]. Weak alignment of requirements engineering with testing may lead to problems in delivering the required products in time with the right quality. For the problem of what features to deliver in a release within given constraints (also called strategic release planning), several empirical studies have been reported in a systematic literature review performed by Svahnberg et al. [9]. In their review, out of 28 articles on strategic release planning, only two take quality constraints into account.

For the problem of when to release features, fewer studies have been published. Decisions about when-to-release have to balance the potential competitive advantage through faster delivery needs against the degree of readiness of the product and the added value through new and revised features. Ho et al. [10] propose an approach that determines a set of trade-off solutions related to the value of the implemented features, their estimated quality, and the release time. But the authors only consider the cost of quality of a feature as a single value in direct relation with testing effort.

Lindgren et al. [11] propose a capability model for improving the release planning process of an organization based on empirical data from a multiple industrial case study. Their model consists of three different process activities, i.e. elicit needs, make release decisions and realize needs. Release planning in context of our industrial case study comprises only release decisions making as the other two activities are covered by requirements elicitation and development as well as testing activities, respectively. For the release decision making activity, the investigated industrial process covers all key areas identified by Lindgren et al., i.e. decision material, product strategy and release plan decision. In addition, our case study takes quality aspects into account for all of these.

Although a significant portion of reported release planning approaches is evaluated in industrial context [9], it is still rare to find release planning approaches that are thoroughly evaluated in full scale industry study [12]. Only few publications empirically analyze industrial release planning processes informally, which are the majority of cases. Du et al. [13] present two empirical studies to compare informal and automated optimization-based planning of software releases. The understandability of the computer-based solutions was improved when adding more explanations to them. Zorn-Pauli et al. [12] analyze an industrial strategic release planning process in an industrial case study and present lessons learned to improve and customize the process in practice. In particular, a requirements abstraction and solution model to support feature generation is provided. But neither Zorn-Pauli et al. nor any other approach empirically analyze quality aspects based on open and closed defects as well as testing aspects in industrial release planning processes.

3. BACKGROUND

In this section, we present background on what-to-release, when-to-release and quality-driven release planning as well as on the proposed solution approach Q-EVOLVE II. Finally, we present the studied research questions.

3.1 What–To-Release Planning

A product feature is a set of logically related requirements that provide a capability to the user and enable the satisfaction of business objectives [1]. As part of any incremental and iterative software development, the question which new features should be offered in upcoming releases is of key importance for product success. From an existing pool of features, the selection and schedule decisions are highly information intensive. To decide about the features, and assign them to one of the upcoming releases, requires a good understanding of the features, their market value, their mutual dependencies and synergies, their effort to get them implemented and the market needs and trends to select the most appropriate one for enhancing or updating the existing product. The main difficulty of the problem is that a large portion of the requested information is continuously changing.

We apply here a variant of the analytical and optimization base approach EVOLVE II and its tool implementation called ReleasePlanner™. The main capability of the system is to conduct intelligent search as part of various planning scenarios. From looking at a sequence of problem-solution pairs for varying parameters, the user can expect a better insight into the problem-solving situation. Optimized planning alternatives are generated which serve as a guideline for human decisions. The plans are optimized in towards best stakeholder satisfaction utilizing the resources available in a best possible way.

3.2 When-To-Release Planning

The when-to-release problem [10] is defined as the process of planning for an upcoming release as a proactive, analytic analysis of trade-off solutions between the product release date, release readiness and release value.

Reliability is one key attribute to decide about the release date. It is often measured based on remaining defects in the system. As software systems get more complex, completely removing all defects is very difficult. Engineering "Just Right" Reliability is very important as the budgets are limited, the competition in the market is intense, risk is high and the time to market has to be met. "More reliability" is not necessarily the only goal; there must be a "balance" between the customer's often conflicting needs for reliability, cost, risk, available budget and software release date.

Testing comprises of activities that will increase the release readiness of the software product, such as testing to discover defects and fixing of such defects. Testing and predicted reliability is studied extensively by software reliability growth models [18]. The models, based on statistical analysis of real-world projects, indicate that the defect-detection rate decreases as the number of defects detected increases, and the total number of defects detected asymptotically approaches a finite value. This would mean that (1) the release is getting more stabilized as more effort is invested in testing, and (2) after a certain point in time in release cycle, even extended testing will only discover minimal number of new defects. Based on this model, testing effort (for low-priority features) can be traded for effort to implement features. Given the estimated testing effort and the estimated effort required to build new features, by defining and analyzing diverse release scenarios, experts can then make the selection between the various trade-off solutions generated.

3.3 Quality-Driven Release Planning

The traditional view of release planning favors delivery of just functionality, thereby largely ignoring quality of the whole product. While quality is a multidimensional phenomenon, we restrict it here to the aspect of achieving zero defects. For a feature to become part of a product release it needs to be implemented and thus, it consumes resources. The core question at this point is how to utilize the resources in the best possible way to provide the best combination of pieces of functionality. However, what is completely lacking in this view is the quality aspect of the resulting release product(s).

Depending on the granularity of the planning, the specific effort attributed to higher target levels of quality can be feature related and/or related to the whole product release (as a form of cross-cutting concern). With this formulation given, the problem is no longer just finding a plan to implementing the most comprehensive and attractive set of features. Instead, the problem now becomes a trade-off analysis where the concern is to provide balance between the most comprehensive and attractive functionality in dependence of varying levels of target quality. A prototype tool for supporting release planning of quality requirements and its initial industrial evaluation has been presented in [14].

3.4 Quality-Driven Approach Q-EVOLVE II

Our proposed solution approach consists of two phases. Phase 1 performs what-to-release planning, and the subsequent Phase 2 is devoted to a more fine-grained when-to-release planning solution. Both phases are formulated as the evolutionary problem solving process contains three phases called *Modeling*, *Exploration*, and *Consolidation* called *EVOLVE II* [1]. The whole process is implemented as the tool called ReleasePlanner™ [16] in combination with the recent plugin called *W2RP* [10].

Modeling includes the definition of a list of candidate features as well as their dependencies and constraints. Modeling also includes the description of what is, or contributes to, the optimality of a solution and which resource or budget constraints need to be fulfilled. Other data, such as stakeholder evaluation of features, are considered to be part of modeling as well. Besides looking at the different features just from the priority perspective, it is important to consider them from the perspective of their resource consumption as well. In reality, the necessary resources needed to implement all features within a given time frame are almost never available.

For performing the step of *Exploration*, a wide range of methods and techniques exist, that have proved themselves useful in very different contexts. The different optimization techniques including exact as well as heuristic approaches fit into this category. Meta-heuristics such as the ones based on genetic algorithms or simulated annealing are receiving more and more attention because of their broad application success. But Exploration is not limited to these techniques. If appropriate, reasoning, machine learning or simulation is applicable whenever it contributes to problem solving.

The *Consolidation* step brings in the knowledge and experience of the human experts. This involvement is aimed at addressing implicit constraints, which are hard or impossible to handle in a completely formalized manner. Consolidation refers to the process of evaluating the qualified solution alternatives, selecting the best (compromise) solution or making suggestions for refinement of the underlying models. An improved understanding of the problem typically results in modifying parts of the underlying model. It also means we have found a formulation of the problem, which is closer to the actual real-world problem.

In more detail, Phase 1 consists of 13 steps which are described in detail in [1]. As a result, five optimized and mutually diversified release plan strategies are generated. For all of them, utilization of resources is optimized against the stated planning objectives.

Phase 2 is applied for each of the releases defined from Phase 1. A more fine-grained consideration including quality is performed, as illustrated in Figure 1. Starting from the release, defined as Baseline release x_0 (RD_0, F_0), with the expected release date RD_0, and the set of features applied as F_0. From varying resource allocations, different trade-off solutions x_i, balancing between release date RD_i, business values in the term of F_i, and quality indicator, are generated and offered to the human experts and decision-makers.

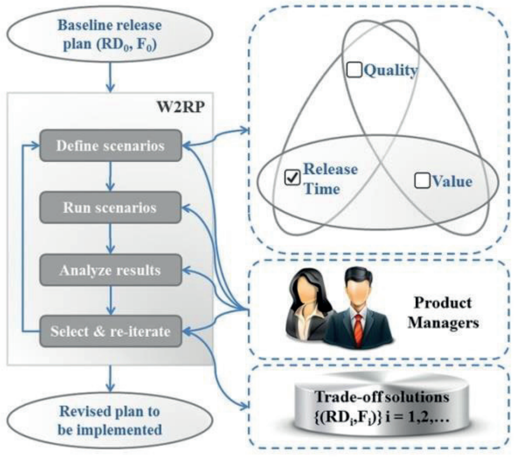

Baseline release plan (RD_0, F_0)
W2RP
Define scenarios
Run scenarios
Analyze results
Select & re-iterate
Revised plan to be implemented
Quality
Release Time
Value
Product Managers
Trade-off solutions {(RD_i,F_i)} i = 1,2,...

Figure 1. W2RP Workflow for Fixed Release Date

3.5 Research Questions

The core content of our explorative research is investigating the impact of combining (1) systematic and data-driven release planning with special focus on (2) quality-driven release planning. From an empirical perspective, we consider two treatments:

Treatment A: Release planning in an ad hoc fashion and without consideration of quality aspects (defects).

Treatment B: Systematic and data-driven release planning with special focus on quality-driven release planning (application of Q-EVOLVE II).

Treatment A is the real-world project having been performed in a public health insurance institution in Austria. Treatment B is conducted retrospectively, using the same data, but applying Q-EVOLVE-II. We propose a customized approach utilizing the key steps of the EVOLVE II method [1]. We call the solution generated from Treatment A the baseline solution. In comparing both treatments, we investigate the following three research questions:

RQ1: Is the application of Q-EVOLVE II able to propose plans to implement the same amount of features in less time than the baseline solution?

RQ2: Is the application of Q-EVOLVE II able to propose plans to implement, in the same time, a larger amount of functionality than the baseline solution?

RQ3: Is the application of Q-EVOLVE II able to propose plans to implement, in the same time, the same amount of functionality, but with less defects slipping through than for the baseline solution?

4. CASE STUDY DESIGN

Case studies are an established empirical method aimed at investigating contemporary phenomena in their context [15]. We study quality-driven release planning in the industrial context of the release planning process of a public health insurance institution in Austria.

4.1 Industrial Context

In the studied institution, a standard development process is in place, consisting of the phases of (1) analysis and design, (2) development, (3) testing, and (4) operating and maintenance. All products are developed in an iterative and incremental development process performing phases (1), (2), and (3) in each release cycle, thereby incorporating the results from an ongoing release in the definition of the next release, before going into operation in phase (4). The test process is based on the standard test process of the International Software Testing Qualifications Board (ISTQB) [6] and comprises the phases test planning, design, execution and evaluation.

Product P is a web application for refunding invoices of medical care for insured persons and managing these cases. The characteristic of Product P is summarized in Table 1. The total effort of Product P is 50 person years and the maximal effort of system testing per release is roughly 40%. In addition, 400 users had to be trained before go-live and about 25 management board meetings as well as numerous jour fixes with developers and testers were held. The final release was deployed in February 2014.

Table 1. Characteristics of Product P

Characteristics	
Area	Web application for refunding invoices of medical care and managing these cases
Staff	43
Number of releases	5
Number of features	21
Number of requirements	250
Number of system tests	2,500
Total effort	50 person years
Maximal effort of system testing per release cycle	40% of overall product effort

4.2 As Is Release Planning Process

The process of release planning in the studied institution has five steps shown in Figure 2. The key stakeholders involved in these are the analyst, project manager and test manager. The project manager is also responsible for product management. Furthermore, additional input for the assessment of features is provided by domain experts, developers, and system testers. On the basis of a project definition and the assessed features, the release planning process steps are performed. To each step, we annotate the involved key stakeholders and its input and output artifacts.

Step 1: Definition of Releases

On the basis of the project definition covering its scope as well as time, budget and quality constraints, the project manager specifies – supported by the analyst - the number of releases and the capacity for each of them. In general, the goal is to deliver the product with all its features meeting the time, budget and quality constraints. The resulting basic release definition may have to be altered in the course of a more detailed analysis in Step 2.

Step 2: Assignment of Features to Releases

Supported by the project manager, the analyst has to partition the features of the product which are pre-assessed by the involved stakeholders and each related to a bundle of requirements. It is considered that dependencies between the different features in the resulting release plan are minimized to avoid redesign of precedent releases.

Step 3: Estimation of Effort and Duration of Actual Release

The project manager and the test manager estimate the effort and duration for testing and development of the actual release taking the release plan into account. The estimation is based on the experience of the managers and results in a concrete project plan for a release defining its development and test schedule as well as resources. The project plan has also to be presented to the management board.

Step 4: Monitoring of Development and Testing Activities

The development and testing activities are monitored by the project and test management based on the project plan and test reports reflecting new and closed defects. A key issue in monitoring release plans is to assess the quality of a release and to recommend which defects should be corrected in the next release or could be postponed. These recommendations are collected in management reports.

Step 5: Decision on Deployment of Next Release

Decisions when and what to deploy in the next release are made in change control board meetings where the project manager, the analyst and the test manager are involved. These decisions are made ad hoc and take the management reports as well as the capacities into account. Based on the agreement the next release is estimated in Step 3 or the release process terminates if the product is finally released.

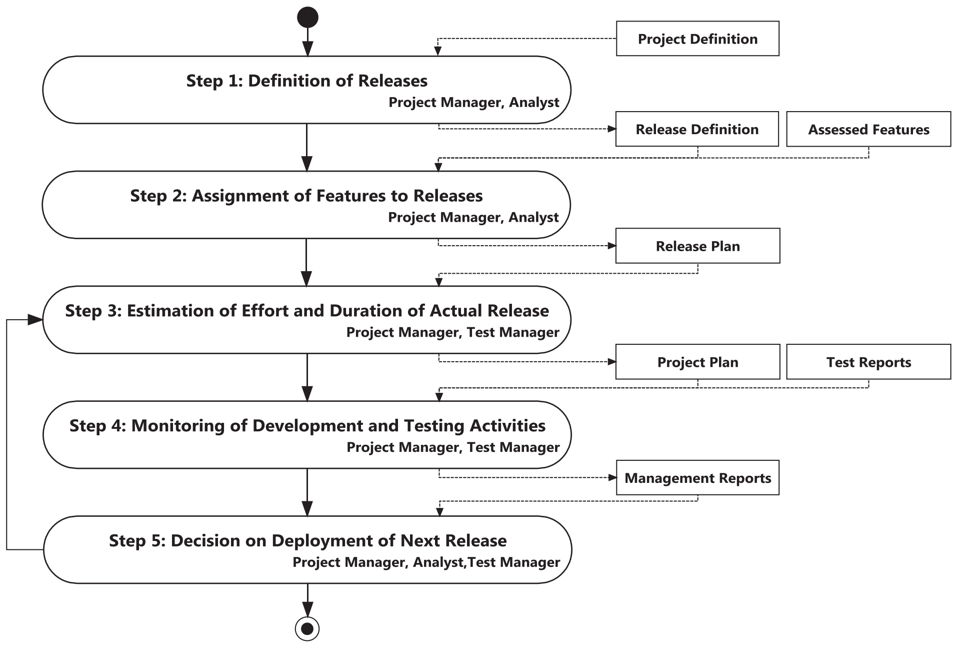

Project Definition
Step 1: Definition of Releases
Project Manager, Analyst
Release Definition
Assessed Features
Step 2: Assignment of Features to Releases
Project Manager, Analyst
Release Plan
Step 3: Estimation of Effort and Duration of Actual Release
Project Manager, Test Manager
Project Plan
Test Reports
Step 4: Monitoring of Development and Testing Activities
Project Manager, Test Manager
Management Reports
Step 5: Decision on Deployment of Next Release
Project Manager, Analyst,Test Manager

Figure 2. Release Planning Process Steps and Artifacts of the Studied Institution

The above release planning process is based on the institution's release planning process and has the following specifics. The release duration defined in the project definition was about 4 months. Already in the first step, the project manager and the test manager estimated effort and duration of each release cycle based on expert estimation and created a project plan. In the third step, the effort and the duration of development and system testing, which, in case of Product P, includes system integration testing of the interface to another product, was estimated for all releases based on the availability of developers and testers.

In general, in Product P, decisions on when and what to release were made ad hoc and the implementation of functionality was ranked higher than quality aspects: The development of new features and change requests was prioritized over bug fixing which was shifted to later releases. Also additional requests shortly before release were taken into account at the expense of testing. Planning was driven by the goal of the project manager to balance the workload of the developers, without taking into account release quality and testability.

4.3 Data Collection Procedure

The data collection in the actual release process of Product P was performed by the test manager who had complete access to the project and product data together with one researcher.

The actually planned features per release were extracted from the project definition. The development effort for each feature was extracted from development reports. Features were assessed based on four criteria, i.e. impact, testability, reliability, and value of testing by stakeholder's project management, domain experts, analysts, test management, development and system testing. The capacities for feature development were extracted from the project plan. The test effort (except test planning effort which is part of project planning) consists of test design and test execution effort. Latter includes the effort for test evaluation. The effort for test design was estimated based on the number of test cases and past experience. The test execution effort was done based on test effort reports. Bugs and their severity were extracted from the defect management tool. The effort for bug fixing was estimated based on the severity type and project experiences by the test manager.

For further details, all the collected data of this study is made available online[1].

4.4 Tool Support

For the planning process, the baseline solution was created ad hoc and manually, without any further tool support. For the subsequent retrospective analysis, the proprietary decision support tool ReleasePlanner™ [16] was used. Based on the Q-EVOLVE II method outlined in Section 3.4, the main capability of the system is to conduct intelligent search as part of various planning scenarios. From looking at a sequence of problem-solution pairs for varying parameters, the user can expect a better insight into the problem-solving situation. Optimized planning alternatives are generated which serve as a guideline for human decisions. The plans are optimized in towards best stakeholder satisfaction utilizing the resources available in a best possible way.

Based upon stakeholder priorities and the estimated resource consumption of the candidate features, the system generates a set of optimized release plan alternatives. Specialized integer programming algorithms exploiting the special structure of the problem are applied. These algorithms are applied in combination with heuristics to speed up the process. Different runtime options are offered to balance between computational time and accuracy of results. For the solutions generated, the algorithms are able to provide a guaranteed level of optimality.

After the completion of this step, each alternative is utilized as an input for the W2RP plugin tool [11] to created different trade-off solutions as indicated in Phase 2. The trade-off solutions are then presented in chart form, with H (RD, Value, Quality) as each of the solution's coordinate. Experts can then view, analyze and pick the most optimized solution, with consideration to the best combination of timeline, features and testing.

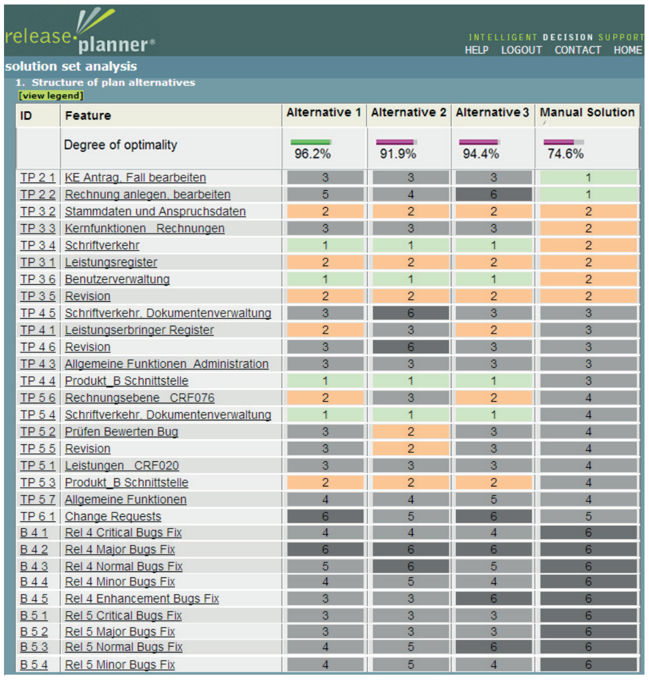
4.5 Baseline Solution

As presented in Figure 3, the manual plan for Product P went through 4 main releases, with ID from TP_2 to TP_5. Change requests (TP_6) were not implemented due to the shortage of time and business deadlines. Based on this release plan, with reference to Figure 4, the company exceeded the available resources, e.g. 306.9% resource required in earlier releases, yet underutilized resources in later releases, e.g. less than 50% resource utilization in Release 3 and 4. This means that developers either had to put in more work hours to complete the assigned tasks, or the features delivery will be delayed due to inaccurate estimates of capacity and effort consumption.

Table 2. Defects Data for Baseline Release Plan

Release	New Defects	Closed Defects	Overall Open Defects	Critical & Major Defects
Release 1	104	1	103	
Release 2.1	172	40	235	
Release 2.2	76	153	158	
Release 3.1	280	134	304	79
Release 3.2	149	97	356	17
Release 4	82	261	438	45
Release 5	159	37	**597**	**72**

Furthermore, as presented in test data in Table 2, there are many defects that were discovered, and several critical and major defects that remained not fixed at the point of release. At the end of the final release, there are still 597 open defects in the system, with 72 (12%) in critical and major category. This resulted in high potential risk and low reliability for the release of the software.



ID	Feature	Alternative 1	Alternative 2	Alternative 3	Manual Solution
	Degree of optimality	96.2%	91.9%	94.4%	74.6%
TP 2 1	KE Antrag, Fall bearbeiten	3	3	3	1
TP 2 2	Rechnung anlegen, bearbeiten	5	4	6	1
TP 3 2	Stammdaten und Anspruchsdaten	2	2	2	2
TP 3 3	Kernfunktionen Rechnungen	3	3	3	2
TP 3 4	Schriftverkehr	1	1	1	2
TP 3 1	Leistungsregister	2	2	2	2
TP 3 6	Benutzerverwaltung	1	1	1	2
TP 3 5	Revision	2	2	2	2
TP 4 5	Schriftverkehr, Dokumentenverwaltung	3	6	3	3
TP 4 1	Leistungserbringer Register	2	3	2	3
TP 4 6	Revision	3	6	3	3
TP 4 3	Allgemeine Funktionen Administration	3	3	3	3
TP 4 4	Produkt_B Schnittstelle	1	1	1	3
TP 5 6	Rechnungsebene CRF076	2	3	2	4
TP 5 4	Schriftverkehr, Dokumentenverwaltung	1	1	1	4
TP 5 2	Prüfen Bewerten Bug	3	2	3	4
TP 5 5	Revision	3	2	3	4
TP 5 1	Leistungen CRF020	3	3	3	4
TP 5 3	Produkt_B Schnittstelle	2	2	2	4
TP 5 7	Allgemeine Funktionen	4	4	5	4
TP 6 1	Change Requests	6	5	6	5
B 4 1	Rel 4 Critical Bugs Fix	4	4	4	6
B 4 2	Rel 4 Major Bugs Fix	6	6	6	6
B 4 3	Rel 4 Normal Bugs Fix	5	6	5	6
B 4 4	Rel 4 Minor Bugs Fix	4	5	4	6
B 4 5	Rel 4 Enhancement Bugs Fix	3	3	6	6
B 5 1	Rel 5 Critical Bugs Fix	3	3	3	6
B 5 2	Rel 5 Major Bugs Fix	3	3	3	6
B 5 3	Rel 5 Normal Bugs Fix	4	5	6	6
B 5 4	Rel 5 Minor Bugs Fix	4	5	4	6

Figure 3. Release Alternatives and Manual Solution

[1] http://homepage.uibk.ac.at/~c703409/RP_ProductP.xls

5. CASE STUDY ANALYSIS

In this paper, we study the impact of applying Q-EVOLVE II for the case study project described in the former section. The results are available online as an Excel spreadsheet[2].

The description of the results is organized according to the research questions RQ 1 to 3 as stated above. In the sequence of the analysis, four different plans shown in Figure 3 are compared with each other:

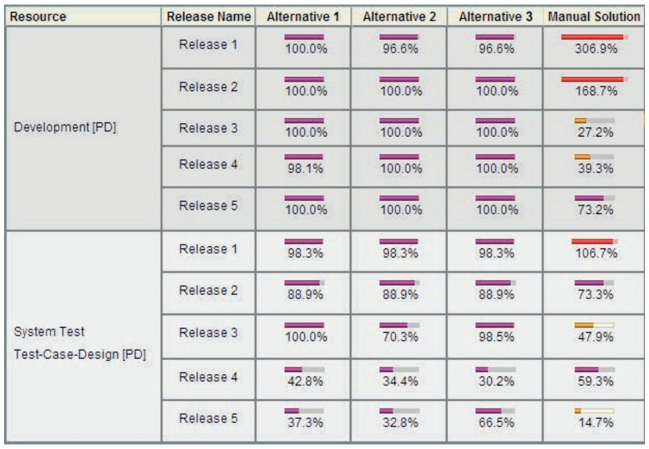
- *Manual Solution* is the actual release process of Product P. This is the baseline plan, annotated as x_0, used for comparison.
- *Alternative 1* presents the plan x_1 explained further in 5.1 is the plan that offers the same features in shorter timeframe.
- *Alternative 2* presents the plan x_2, explained further in 5.2 is the plan that offers more feature value in the same timeframe.
- *Alternative 3* presents the plan x_3, explained further in 5.3 is the plan that offers the improvement in quality based on testing.

5.1 RQ1: Release the Same Features in Shorter Time

Let $F_1(x)$ be the total effort (in person days) to implement release plan x. As the effort is measured in person hours, the reduction in resource consumption can translate directly to shortened release time frame. The manual plan x_0, including the overestimate for Release 1 and 2, consumes 1232 person hours. Plan x_1 only required 510 person hours, as calculated from the data spreadsheet. Alternative x_1 is achieved with just 41% of the resource consumption of the baseline solution.

$$F_1(x_1)/F_1(x_0) = 41\% \qquad (1)$$

We interpret this as that the same amount of functionality can potentially be released in 59% units of time earlier. Feedback from the project and test manager of Product P appreciates Alternative 1. It is especially highlighted, that critical features TP_4_4 and TP_5_4 are relocated to the first release, thus mitigating the risk of these problematic features for the final product release.



Resource	Release Name	Alternative 1	Alternative 2	Alternative 3	Manual Solution
Development [PD]	Release 1	100.0%	96.6%	96.6%	306.9%
	Release 2	100.0%	100.0%	100.0%	168.7%
	Release 3	100.0%	100.0%	100.0%	27.2%
	Release 4	98.1%	100.0%	100.0%	39.3%
	Release 5	100.0%	100.0%	100.0%	73.2%
System Test Test-Case-Design [PD]	Release 1	98.3%	98.3%	98.3%	106.7%
	Release 2	88.9%	88.9%	88.9%	73.3%
	Release 3	100.0%	70.3%	98.5%	47.9%
	Release 4	42.8%	34.4%	30.2%	59.3%
	Release 5	37.3%	32.8%	66.5%	14.7%

Figure 4. Development and Test Design Resource Consumption of Release Alternatives

Plan x_1 also improves the balance of development and testing resources as shown in Figure 4. This is an important issue for the project and test management. The freed up resources allow shifting the release focus purely from feature implementation to also consider bug fixing from the very beginning, which reduces effort as well as the risk of meeting the deadline and increases quality. Especially fixing of critical bugs in release 4 maximizes release quality and avoids time and effort wastage in system test.

[2] http://homepage.uibk.ac.at/~c703409/RP_ProductP_Results.xls

5.2 RQ 2: Release More Feature Value in the Same Time Frame

As illustrated from Figure 3, alternative plan x_2, which is released at the same time as the baseline, can include change requests, which are voted as high priority by both stakeholders and product managers. Let $F_2(x)$ be the stakeholders' features scores of the release plan, indicating the potential of the plan to increase values. Then, the relative improvement can be determined by the following ratio.

$$F_2(x_2)/F_2(x_0) = 130\% \qquad (2)$$

Alternative 2 releases 30% more feature value than the baseline plan during the same time frame. Additionally, Figure 3 shows, that in Alternative 2 considerably more bugs can be fixed before the final release, i.e. Release 5, than in the manual baseline solution x_0 resulting in higher release reliability [15] of Alternative 2 as well.

5.3 RQ 3: Release Better Quality in the Same Time Frame

As illustrated in Figure 3, alternative plan x_3 includes the defect discovery and critical, major and minor bug fixing for all releases. We utilize the estimates of effort required for test execution and defect fixing based on severity in Table 3. According to this plan, more testing is executed, and more defects are fixed, which means higher release reliability [15], as explained in Section 2. Let $F_3(x)$ be the total number of defects remaining in the product after being released. Based on the case data, when testing and fixing are executed, only 87 defects remained in the system, compared to 597 having been there originally.

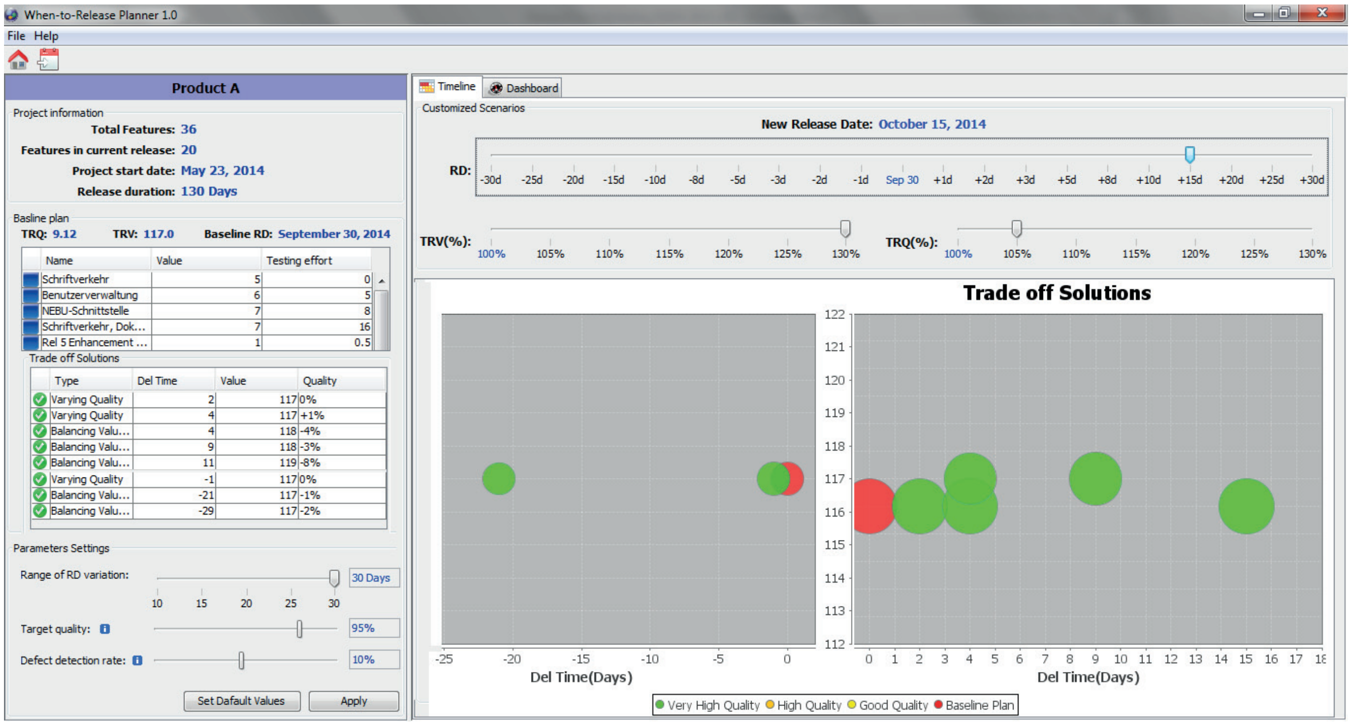
$$F_3(x_3)/F_3(x_0) = 87/597 = 15\% \qquad (3)$$

This means, there are 85% less defect slipping through following plan x_3. From the test management point of view, it was stated that following x_3 bug fixing is distributed towards the different releases and considered relatively early in the release development, which has benefits for the workload of the test team.

5.4 RQ1, RQ3: When-To-Release Trade-Off Analysis for Better Quality in Shorter Time

Further to the above highlighted cases, utilizing results from the W2RP problem described in [16] and in Phase 2, we executed an interactive simulation of when-to-release for all alternative plans generated. Figure 5 below describes the potential trade-off outcomes for Alternative 1 that can be achieved by varying the release date and different types of resources distribution. Potentially, product P can be released up to 30 person days earlier by readjusting just 2% of testing and fixing effort for the existing test profile, which is already superior to the testing execution that was done on the baseline release x_0.

Alternatively, by allowing another 11 days, and a compromise of 8% in quality, there are more features that can be offered, bring the value up to 119 (versus 117). This result is very useful for the project and test management of product P as it allows releasing Product P earlier without expected loss in quality. This result contributes to both RQ1 and RQ3 as product P can be released with the same features as baseline in less time and consecutively, allowing some testing and fixing of defects. Thus, the when-to-release analysis helps in the support of risk management.


When-to-Release Planner 1.0

Product A

Project information

Total Features: 36

Features in current release: 20

Project start date: May 23, 2014

Release duration: 130 Days

Basline plan

TRQ: 9.12 **TRV:** 117.0 **Baseline RD:** September 30, 2014

Name	Value	Testing effort
Schriftverkehr	5	0
Benutzerverwaltung	6	5
NEBU-Schnittstelle	7	8
Schriftverkehr, Dok…	7	16
Rel 5 Enhancement …	1	0.5

Trade off Solutions

	Type	Del Time	Value	Quality
✓	Varying Quality	2	117	0%
✓	Varying Quality	4	117	+1%
✓	Balancing Valu…	4	118	-4%
✓	Balancing Valu…	9	118	-3%
✓	Balancing Valu…	11	119	-8%
✓	Varying Quality	-1	117	0%
✓	Balancing Valu…	-21	117	-1%
✓	Balancing Valu…	-29	117	-2%

Parameters Settings

Range of RD variation: 30 Days

Target quality: 95%

Defect detection rate: 10%

Customized Scenarios

New Release Date: October 15, 2014

Figure 5. Release Alternatives in Shorter Release Time with Minimal Compromise to Testing and Reliability Activity

However, the when-to-release analysis results presented through the tool, were less understandable to the decision makers than the what-to-release results. More effort is required to explain effectively, which is supported by the interactive features of the W2RP plugin.

5.5 Threats to Validity

Threats to validity of empirical research have to be examined during all phases of the case study. To evaluate the validity of this case study, the validity perspectives proposed by Wohlin et al. [17] were considered and are analyzed in the following. The threats to construct validity are reduced by cooperation with the industry partner over more than one and a half year. The consistency of the results gained from retrospective analysis, i.e. improved quality and/or more features offered, was assured by running several simulations and presenting the outcomes to the practitioners for review.

The used data was collected by the directly involved practitioner and one researcher. The two other researchers reviewed the data. Further, the threats to internal validity were reduced by using an established tool, i.e. ReleasePlanner™ [16] for our analysis. This tool was validated in many academic and industrial projects. Finally, the threats to external validity are reduced by conducting the case study in a real-world industrial setting and by integration of practitioners.

6. DISCUSSION

During the life project for product P, decisions on when and what to release were made ad hoc. Implementation of functionality was prioritized over product quality with resulted in a larger number of defects detected after delivery. During development, bug fixing had to be postponed several times due to missing capacities and relatively high number of defects in general caused additional test effort.

Product P is now in operation and in the meantime eight bug-fixing releases were deployed. As testing and bug fixing was not considered appropriately from the very beginning, tests could not be executed in later phases leaving test resources and in further consequence also development resources unexploited and making release planning even less structured and ad hoc. In addition, also several external services had to be used with limited system integration testing.

As demonstrated in the retrospective scenarios, systematic and analytical release planning with Q-EVOLVE II helps improving the quality of the released product, and potentially offers more features or change requests to the final product. Predictive analysis also allows users to have a holistic view of different release alternatives, together with their projected impact on resources, optimality, and quality of the final outcomes. Although, the release planning process itself is often complex and formulating an optimization problem is hard [11], Q-EVOLVE II turned out to be applicable and to improve release planning in the studied institution.

The release planning data was prepared and managed in spreadsheets which are already used for other project management tasks. Feedback from practitioners stated the usefulness of the automatically generated release alternatives to support release decisions. But it was also highlighted that manual adaptations to these plans may be required. Practitioners favored release alternative 1 of Figure 3 as it “mitigates the risk of not meeting the required quality standards and the aim of the project management to meet the deadline for rollout of the product P if features TP_4_4 and TP_5_4 are re-assigned manually”.

Quality-driven release planning with Q-EVOLVE II can be utilized to improve the actual release planning process of the institution in general reducing the risks of not meeting the deadline and poor quality by critical bugs slipping through.

The assignment of features to releases in Step 2 of the release planning process in the studied institution can be supported by strategic release plans generated in Phase 1 of Q-EVOLVE II. These plans (see Figure 3) provide Pareto-optimal assignments of features to releases taking capacities into account. Based on the objectives of the planned product and especially when taking quality aspects into account, the project manager can select one of several release plan alternatives. However, in practice the automatically generated release plans have to be adapted manually to fulfill all project constraints.

Due to the improved utilization of resources in the provided release plans, also the estimation of effort and duration in Step 3 may be easier. The release duration and planning of milestones has to be defined in a project plan. Because these plans have to be presented and argued also to the management, additional information as provided by the generated release alternatives, e.g. the degree of optimality, and W2RP are more than helpful. Especially the trade-off between release time and quality provided by W2RP supports concrete decisions on the release date taking quality into account. But it turned out that the output of W2RP as shown in Figure 5 requires additional explanation to be understandable to decision makers. For instance, simulations provided by W2RP, explanation of the underlying formulas, and also linking the output of W2RP to concrete defect trend analysis data of new and resolved defects per release could make arguments more transparent.

Finally, also decisions of the deployment of the next release in Step 3 benefit from Q-EVOLVE II as adapted release plans can be generated to support further decisions. For instance, the retrospectively generated release alternative 1 could support the test manager to argue for more testing and bug fixing as this can increase the degree of optimality and the resource workload compared to the baseline.

7. CONCLUSION AND FUTURE WORK

From comparing the actual project performance with several retrospective scenarios, we have demonstrated that quality-driven release planning can provide different types of advantages. Being explorative in nature, we consider the value of this research in outlining the methodology (Q-EVOLVE II) and demonstrating its impact and applicability. These results are a confirmation and follow-up to former controlled experiments done to compare ad hoc with tool supported and analytical release planning [13]. Similarly, quality-driven release planning considering feature implementation at different levels of functionality and quality is getting more attention [7].

Further case study and other investigations are needed in order to claim external validity of the findings. It will be interesting to see the impact of the proposed process changes in a subsequent project. More detailed and comprehensive data collection procedures will allow performing the various scenarios more reliably. Finally, future investigations have started to investigate the range of opportunities available to get access to distributed knowledge and information and its utilization for the release planning process [19], resulting in new treatments to be considered. Open innovation as a beneficial approach for companies and users integrates internal and external ideas and paths to market at the same level of importance and distributed talent, knowledge and ideas into innovation processes.

ACKNOWLEDGEMENTS

This work was sponsored by the projects QE LaB – Living Models for Open Systems (FFG 882740) and MOBSTECO (FWF P 26194-N15) as well as by the Natural Sciences and Engineering Research Council of Canada, NSERC Discovery Grant 250343-12. In addition, we are grateful to Maleknaz Nayebi for her review and suggested improvements.

REFERENCES

[1] G. Ruhe. Product Release Planning. CRC Press, 2010.

[2] T. Gorschek, S. Fricker, K. Palm, and S. Kunsman. A lightweight innovation process for software-intensive product development. IEEE Software 27, 37–45, 2010.

[3] B. Boehm and V. R. Basili. Software Defect Reduction Top Top 10 List, 34(1):135-137, 2001.

[4] C. Denger and T. Olsson. Quality assurance in requirements engineering, Engineering and managing software requirements. pp. 163-185, Springer, 2005.

[5] C. Ebert and S. Brinkkemper. Software product management - An industry evaluation. Journal of Systems and Software 2014 (in press).

[6] ISTQB. Standard glossary of terms used in software testing. Version 2.2. International Software Testing Qualifications Board, Glossary Working Party, 2012.

[7] E.Bjarnason, P. Runeson, M. Borg, M. Unterkalmsteiner, E. Engström, B. Regnell, G. Sabaliauskaite, A. Loconsole, T. Gorschek, and R. Feldt. Challenges and practices in aligning requirements with verification and validation: a case study of six companies. Empirical Software Engineering, online first, Springer, 2014.

[8] Z. A. Barmi, A. H. Ebrahimi, and R. Feldt. Alignment of requirements specification and testing: A systematic mapping study. Conference on Software Testing, Verification and Validation Workshops (ICSTW 2011), IEEE, 2011.

[9] M. Svahnberg, T. Gorschek, R. Feldt, R. Torkar, S. B. Saleem, B., and M. U. Shafique. A systematic review on strategic release planning models. Information and Software Technology, 52(3):237-248, 2010.

[10] J. Ho, S. Shawanez, and G. Ruhe. A Prototype Tool Supporting When-to-release Decisions in Iterative Development. 2nd International Workshop on Release Engineering (RELENG), 2014.

[11] M. Lindgren, R. Land, C. Norström, and A. Wall. Towards a capability model for the software release planning process - Based on a multiple industrial case study. Product-Focused Software Process Improvement. Springer, 2008.

[12] G. Zorn-Pauli, B. Paech, B., T. Beck, H. Karey, and G. Ruhe. Analyzing an Industrial Strategic Release Planning Process– A Case Study at Roche Diagnostics. Requirements Engineering: Foundation for Software Quality, Springer, 2013.

[13] G. Du, J. McElroy, and G. Ruhe. A family of empirical studies to compare informal and optimization-based planning of software releases. Proceedings ESEM 2006, 2006.

[14] R. Berntsson Svensson, P. Lindberg Parker, and B. Regnell. A prototype tool for QUPER to support release planning of quality requirements, Workshop IWSPM 2011, 2011.

[15] P. Runeson, M. Höst, A. Rainer, and B. Regnell. Case Study Research in Software Engineering: Guidelines and Examples. John Wiley & Sons, 2012.

[16] ReleasePlanner, www.releaseplanner.com, last access on July 5, 2014.

[17] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. Experimentation in software engineering. Springer, 2012.

[18] A. Wood, Software reliability growth models: assumptions vs. reality, Proceedings International Symposium on Software Reliability Engineering. IEEE, 1997.

[19] M. Nayebi, G. Ruhe. An Open Innovation Approach in Support of Product Release Decisions. CHASE 2014 (Co-located Workshop at ICSE 2014), 2014.