University College London
Faculty of Engineering
Department of Computer Science

A Bayesian Approach for Software Release Planning under Uncertainty

Olawole Stephen Oni

First Supervisor:
Dr. Emmanuel Letier

Second Supervisor:
Dr. Earl Barr

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computer Science of University College London
April, 2020

Declaration

I, Olawole Stephen Oni , declare that this thesis titled "A Bayesian Approach for Software Release Planning under Uncertainty" and the works presented in it are my own. I confirm that:

- This work was done mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signature: _____ Date: _____

Abstract

Release planning — deciding what features to implement in upcoming releases of a software system— is a critical activity in iterative software development. Many release planning methods exist but most ignore the inevitable uncertainty of future development effort and business value. The thesis investigates how to analyse uncertainty during release planning and whether analysing uncertainty leads to better decisions than if uncertainty is ignored.

The thesis's first contribution is a novel release planning method designed to analyse uncertainty in the context of the Incremental Funding Method, an incremental cost-value based approach to software development. Our method uses triangular distributions, Monte-Carlo simulation and multi-objective optimisation to shortlist release plans that maximise expected net present value and minimise investment cost and risk.

The second contribution is a new release planning method, called BEARS, designed to analyse uncertainty in the context of fixed-date release processes. Fixed-date release processes are more common in industry than fixed-scope release processes. BEARS models uncertainty about feature development time and economic value using lognormal distributions. It then uses Monte-Carlo simulation and search-based multi-objective optimisation to shortlist release plans that maximise expected net present value and expected punctuality. The method helps release planners explore possible tradeoffs between these two objectives.

The thesis' third contribution is an experiment to study whether analysing uncertainty using BEARS leads to shortlisting better release plans than if uncertainty is ignored, or if uncertainty is analysed assuming fixed-scope releases. The experiment compares 5 different release planning models on 32 release planning problems. The results show that analysing uncertainty using BEARS leads to shortlisting release plans with higher expected net present value and higher expected punctuality than methods that ignore uncertainty or that assume fixed-scope releases. Our experiment therefore shows that analysing uncertainty can lead to better

release planning decisions than if uncertainty is ignored.

Impact Statement

Software systems are most often delivered over multiple releases. Each release improves the previous ones by changing or adding features based on lessons from earlier releases. In this context, release planning is the activity of planning the order in which new features will be developed for future releases. Release planning is critical to manage stakeholders' expectations, maximise business value and organise the software development process. Most release planning methods used in industry ignore the inevitable uncertainty about the time it takes to develop each candidate feature. They also ignore the uncertainty about the future business value of candidate features. As a result, release planning decisions are often based on inaccurate evaluation of candidate features. The objectives of this thesis was to study how to analyse uncertainty during release planning and whether analysing uncertainty leads to better release planning decisions than if uncertainty is ignored.

We have developed a new release planning method under uncertainty, called BEARS, and showed that analysing uncertainty using BEARS leads to shortlisting better release plans than if uncertainty is ignored. Release plans shortlisted using BEARS have higher expected net present value (a common metric to evaluate value flow) and higher expected punctuality (i.e. features are more likely to be delivered on time) than release plans shortlisted by methods that ignore uncertainty. Our new release planning method is supported by a prototype tool that includes all core functionalities for analysing uncertainty during release planning. This prototype provides the basis for developing a novel commercial tool for release planning under uncertainty or to extend existing release planning tools with new features to analyse uncertainty. All software engineering projects that have uncertainty about development time and value would benefit from the techniques developed in the thesis. This includes most software engineering projects but is most relevant to highly innovative projects where uncertainty about development time and business value is higher than for routine projects. The techniques developed in the thesis will help software development teams

have better conversations with their clients and managers about the inherent uncertainty of software development. The techniques allow them to evaluate and communicate the consequences of such uncertainty. The thesis exposes the need for release planners to consider compromises between defining release plans that are over-ambitious (potentially high value but unlikely to be delivered on time) and over-conservative (lower value but easy to deliver on time). Our method helps release planners visualize all optimal tradeoffs between these two extremes and to select a release plans best suited to their preferences.

Acknowledgements

I give thanks and glory to the Almighty God for seeing me through this PhD programme, May his name be praise forever more.

I specially thank my supervisor, Dr. Emmanuel Letier, for his support, guidance, patience and constructive feedback throughout this programme. This research work would not have been successful without his remarkable experience and tutelage. I thank Dr. Earl Barr and Dr. Jens Krinke, my examiners in my first and second year vivas. Your experience, knowledge and critical evaluations have been valuable to this work.

My sincere appreciation goes to my sponsors in Nigeria: The Federal Scholarship Board (FSB), Tertiary Trust Fund (TETFUND) and Presidential Special Scholarship Scheme for Innovation and Development (PRESSID). In addition, I thank the Department of Computer Science for the travel grants awarded to enable me present my research at the 22nd International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ) in Gothenburg, Sweden and the 25th International Requirements Engineering Conference (RE) in Lisbon, Portugal.

Finally, I thank my lovely and beautiful wife, Oyedunni Oni, for her immense supporting throughout my research. I also appreciate my lovely daughter, Hadassah Oni for cooperating with me when I needed to work at home. Special thanks to my parents, Mr & Mrs Oni, for their support from when I started my life journey. Special thanks to my mother in-law Mrs Kofoworola Oyefi for her support and word of encouragement during difficult times in my research. To my siblings, Gbenga Oni, Opeyemi Oni and Titilayo Oni, my parent in-laws, brother-in-laws, sister-in-laws, my spiritual father (Pastor Akintayo Aiyegbusi). Special thanks to my friends Dr. Saheed Busari, Tolulope Agbebi, Olusegun Folarin and Tolulope Adeyelu for their support financially, morally and academically. I thank you all for your support and prayers.

List of Publications

The work presented in this thesis is the original work undertaken between November 2015 and November 2019 at the Department of Computer Science, University College London, United Kingdom. Part of the work presented in this thesis has previously been published in the following venues:

- **Oni O**, Letier E. Optimizing the incremental delivery of software features under uncertainty. *In International Working Conference on Requirements Engineering: Foundation for Software Quality* 2016 Mar 14 (pp. 36-41). Springer, Cham.
- **Oni O.** Towards a Bayesian Decision Model for Release Planning in Incremental Development. *In 2017 IEEE 25th International Requirements Engineering Conference (RE)* 2017 Sep 4 (pp. 520-525). IEEE.

The first publication describes an approach to extend an existing release planning method with uncertainty and multi-objective optimization and forms the basis for the work reported in Chapter 3. The second publication presents early initial work in the development of the release planning method described in Chapter 4. Materials from Chapters 4 and 5 are included in the following manuscript currently under submission:

- **Oni O**, Letier E. Bayesian Economic Analysis for Software Release Planning under Uncertainty. *IEEE Transactions on Software Engineering.* (In Review)

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Iterative software development is a widely recommended approach to build software when requirements are uncertain and the project is subject to significant risks [4, 5]. Instead of delivering all software features at once, the software is delivered over multiple releases where each release adds or modifies features based on stakeholders' feedback and lessons learnt from previous releases. By feature, here, we mean both functional features and quality improvements (e.g. better performance). Since not all features are delivered at once, an essential activity of iterative development consists in prioritizing what features to deliver in upcoming releases. This activity, known as *release planning*, is critical to deliver business value, manage stakeholders' expectation, control risks, and organise the software development activities [6]. Release planning spans the traditional boundaries of requirements engineering, project management and software evolution; it is concerned with deciding what features to build, prioritizing features delivery under time and resource constraints, and incrementally improving the software by adding, modifying or removing features.

Release planning is concerned with decision related to selecting and assigning features to produce a sequence of product releases that satisfies important technical,

resource, budget and risk constraints [6, 7]. A software release is an improved version of an evolving software system characterized by a collection of new or modified features that is valuable to customers [8]. Release planning decisions address the questions of what features to develop in upcoming releases, when to release features and what quality must be satisfied by a software release [7]. For example, in iterative development, the software system is decomposed into features that are valuable to stakeholders. Release planning then involves prioritizing these features using stakeholders perceived value and assigning the features to various releases. The assignment is done in such a way that the most important features are developed first before the less important ones. A bad release planning decision can prove to be costly for a software intensive organization. It can result in loss of valuable customers, waste of organization resources, unsatisfied stakeholders, revenue loss, and loss of market share [6].

Release planning is a complex decisions making problem but crucial to the satisfaction of various stakeholders goals [6, 9]. These decisions largely depends on availability of relevant information about the software system under development, requires good understanding of the software features, business value, dependency relationships, effort, stakeholders' preferences and organisation high level business objectives. Unfortunately, this information are rarely available or complete due to uncertainties. These decisions are further complicated by the complex relationships that exists between the multiple stakeholders viewpoints, competing business objectives, budget constraints and dependencies between features.

1.1 Motivating Example

Throughout the thesis, we illustrate and compare alternative release planning methods using an example based on a real local government project. The project

aims to develop online services allowing local residents and businesses to perform various tasks such as paying local taxes, reporting missed rubbish collection, managing parking permits, submitting and responding to planning applications online instead of over the phone or by visiting offices in person. Many residents and businesses have expressed preferences for performing such transactions online rather than over the phone or in person when possible. Each online interaction is much cheaper for the local government than the equivalent interaction over the phone or in person. The local government has identified over 20 individual services that could be delivered online, but the local government's Information System group has limited resources and is only able to develop a few features supporting these services every quarter. Different stakeholders disagree about which features should be developed first. The project manager and software development team need to prepare a release plan that will manage stakeholders' expectations and organise the development work so that it provides the most value to the local government and its residents. The candidate services to be offered online cover the following areas:

- **Council Tax:** view council tax bills, pay council tax online, set up council tax direct debit, apply for council tax reduction, view council tax reduction claim;
- **Parking:** apply for parking permit, buy visitor parking permit, pay parking fines, contest parking fines;
- **Rubbish Collection:** look up rubbish collection day, report missed collection, order recycling bin;
- **Housing application:** submit a housing application, view housing application status, notify of housing application status change;
- **Planning Application:** submit planning application, view planning ap-

plications, comment on planning applications, create planning application alerts

If the system were to be developed using a non-incremental delivery method, the system will be delivered to the council after all the services have been developed. The danger of this approach is that the council will not get any return on their investment until all the services has been delivered and developers will have no feedback on the effectiveness of the on-line system until it is fully developed and deployed. However, incremental delivery will allow the services to be developed and released in phases thereby ensuring that the council generate early value from the project. Consequently, project managers must decide on the optimal sequence of releasing these services in order to quickly achieve council's cost-saving objective at lowest investment cost and risk.

1.2 State of the Art and Limitations

The scientific and professional literature describes numerous methods to support release planning decisions (see [9, 10] for surveys). These methods involve evaluating candidate features against a set of criteria relevant to the release stakeholders' objectives. Commonly used criteria include development effort measured in person-days or story points, business value measured in abstract value points (e.g. from 0 to 9) or monetary units ($, €, £, ¥, etc.), and stakeholders' satisfaction measured as abstract scores or using problem-specific metrics (e.g. the percentage of user queries responded within a certain time). Different methods propose different sets of criteria, different ways to elicit scores from stakeholders, and different ways to combine the elicited scores into some overall quantity that ultimately informs release planning decisions.

Empirical studies in industrial contexts have shown the many benefits of systematic

release planning methods over ad-hoc release planning methods [11, 12, 13]. By introducing structure to the decision process, release planning methods mitigate problems encountered with ad-hoc planning methods such as the lack of clarity about decision objectives and candidate features, the late identification of feature dependencies, the difficulty of engaging key stakeholders in the decision process, and the uncontrolled interference of power, politics and self-serving interests. Furthermore, the studies show that introducing structure also reduced time and effort involved in making decisions.

Most release planning methods ignore the inevitable uncertainty of software development effort and value [14]. They evaluate candidate release plans as if all features will be delivered on time according to plan. In reality, some features will take longer to develop than predicted and, once released, may deliver more or less business value than anticipated. Ignoring such uncertainty can lead to inaccurate, overoptimistic, and inconsistent evaluation of release plans, which in turn can lead to misinformed and possibly inadequate release planning decisions.

A few methods have been proposed to analyse uncertainty during release planning [15, 16, 17, 18] but they suffer the following limitations:

1. They provide no support for reasoning about uncertainty associated with economic value. Existing release planning approaches that provide uncertainty support only consider effort uncertainty or value point uncertainty. Uncertainty associated with financial business value has largely been ignored.

2. No evidence exists that the added complexity of analysing uncertainty leads to better decisions than simpler methods that ignore uncertainty. Without such evidence, the added complexity cannot be justified and release planning methods under uncertainty are unlikely to be adopted.

3. They rely on assumptions that are incompatible with common current

industrial practices. Surveys of industrial practices indicate that most organisations follow a release process where the release dates are fixed (e.g. every 3 months) but the content of each release is flexible, i.e. the release content may differ from what was planned [13]. The scope of a release is the set of features delivered in the release. Under a fixed-date release process, if development takes longer than anticipated, the release scope will be reduced but the release date will not be changed. However, existing release planning methods under uncertainty assume fixed-scope releases. Under fixed-scope release processes, if development takes longer than anticipated, the release scope will not be changed and the release date will be postponed until all planned features are completed. As a result, existing release planning methods under uncertainty can analyse uncertainty about release dates but not about the release scopes.

1.3 Research Questions

Based on the limitations outlined above, we formulate the following research questions:

1. How can we extend cost-value based release planning methods to analyse uncertainty?

2. How can we analyse uncertainty during release planning in the context of fixed-date release processes?

3. Does analysing uncertainty during release planning lead to better release planning decisions than if uncertainty is ignored? Would different release planning methods applied in the same context recommend the same release plans? If not, do some methods make better recommendations than others?

1.4 Thesis Contributions

The thesis contributions are:

1. Formalization and critical review of existing requirements prioritization and release planning models. We provide a critical review of existing release planning methods identifying the assumptions of these models and discuss the validity of such assumptions (Chapter 2)

2. A novel cost-value based release planning method, called MOIFM, that allows release planners to analyse uncertainty about the development cost and economic value of candidate features in the context of the Incremental Funding Method — an incremental cost-value based approach to software development (Chapter 3).

3. A novel release planning method under uncertainty, called BEARS, that allows release planners to analyse uncertainty about the development time and economic value of candidate features in the context of fixed-date release processes (Chapter 4).

4. Experiments on 32 release planning problems to study whether analysing uncertainty during release planning using BEARS leads to selecting better release plans than if uncertainty is ignored or if one assumes fixed-scope releases. (Section 5).

The thesis will show that analysing uncertainty using BEARS leads to selecting release plans that have higher expected net present value (a common financial metric for comparing value flows) and higher expected punctuality (the percentage of features delivered on time) than if uncertainty is ignored or if the release planning model assumes fixed-scope releases.

1.5 Thesis Structure

The remainder of the thesis is organized as follows:

- Chapter 2 introduces key concepts of release planning and presents a critical review of existing release planning methods.
- Chapter 3 presents a novel release planning method that extends an existing financial method to deal with uncertainty and multiple objectives.
- Chapter 4 presents an overview of a novel release planning framework for release planning under uncertainty when release dates are fixed and work scope is flexible.
- Chapter 5 present empirical studies comparing BEARS with existing release planning methods that ignores uncertainty and methods that reason about uncertainty but uses fixed scope release.
- Chapter 6 presents summary of thesis report and future directions.

Chapter 2

Background on Release Planning

Release planning typically involves planning several months ahead and is performed on coarse-grained work items such as features and epics rather than fine-grained work items such as user stories and specific development tasks [6, 19]. In agile development, release planning is followed by more frequent sprint planning activities [19]. A sprint is a smaller development period (e.g. 2 weeks) within a longer release period (e.g. 3 months). Sprint planning focuses on the next sprint only. Release planning in contrast usually involves planning multiple releases. In this chapter, we will define basic release planning concepts, explain release planning activities, and state of the art release planning approaches.

2.1 Release Planning Concepts

We start by defining the concepts of product backlog, release plans, effort, value, and planning criteria. We also discuss existing approaches to estimate effort in person hours or days and value measured either as abstract value points or in monetary units.

2.1.1 Product Backlog

Release planning takes as input a backlog of work items to be scheduled for future releases and generates as output a release plan specifying what items are scheduled for what future releases.

Definition (Product Backlog): A *product backlog* is

- a set WI of work items that are candidates for future releases; and
- a relation $\leftarrow \subseteq WI \times WI$ that defines a precedence relation between work items, i.e. $w_i \leftarrow w_j$ means that work item w_i must be delivered before or at the same time as work item w_j.

In some release planning methods, backlog items are called requirements [20, 21], in others they are called features [6, 2, 19]. In this thesis, we will follow the terminology of the Incremental Funding Method where work items are either features or architectural elements [2]. Features are functional or quality improvement that deliver value to stakeholders, while *architectural elements* are software elements that do not in themselves deliver value to stakeholders but are prerequisites for the development of valuable features.

Features in the product backlog are primarily identified through analysis of the problem domain, analysis of stakeholders' needs and business wide objectives [6, 21]. Early stages of software development are characterized by diverse uncertainties about system goals and stakeholders' needs. Stakeholders do not often know what they need or how to communicate those needs due to lack of adequate information at the early stage of the project. The process of eliciting these features from stakeholders is a non-trivial *requirements engineering* process [22]. Precedence relations among features in the product backlog are also identified and communicated during the early phase of requirements gathering.

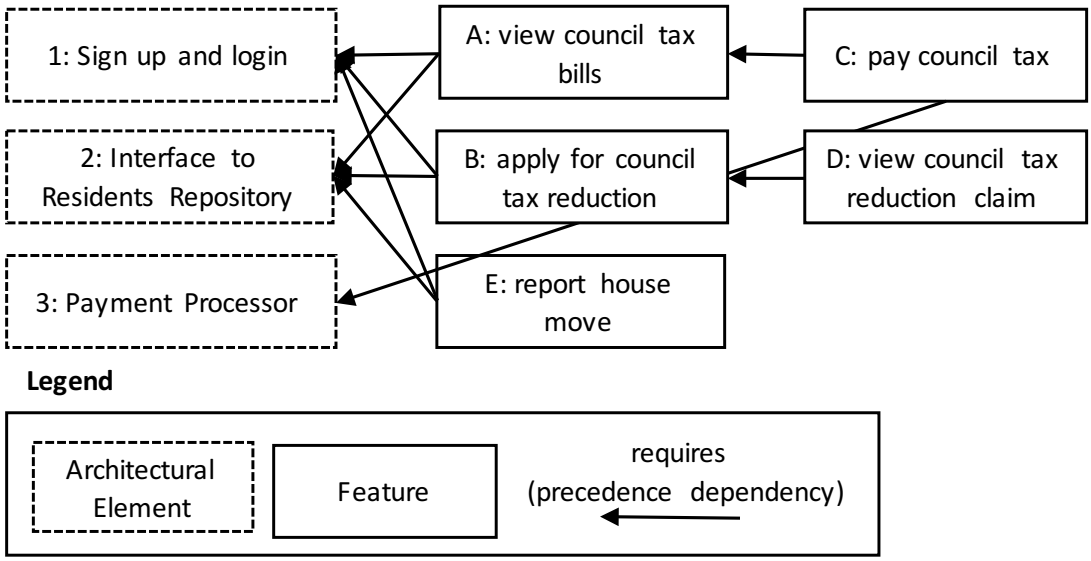


Figure 2.1: Part of the product backlog for the local council project.

Precedence relations between features determine the relative ordering of delivering the features to the market [23]. In addition to precedence relation, the product backlog may also include other types of dependencies between work items such as coupling, exclusion, and weak precedence [6, 24]. We will not use these additional dependencies in this thesis but all techniques described in the thesis can be extended to handle additional dependency types.

As an example, Figure 2.1 shows a small fragment of the product backlog for our local government project described in Section 1.1. The product backlog is composed of 19 features, 6 architectural elements, and 28 dependency links.

2.1.2 Release Plan

The main output of release planning is a release plan.

Definition (Release Plan): Given a product backlog $< WI, \leftarrow >$, a release plan is a partial function $p : WI \rightarrow [1..H]$ that maps work items to future release periods. The number of release periods $H \in \mathbb{N}^+$ in the release plan is called the *planning horizon*. For a work item w_i, $p(w_i) = h$ means that w is planned to be delivered at

the end of the h^{th} release period. A release plan is a partial function because not all work items in the backlog need to be planned for some future period. Release plans must satisfy the precedence constraints: if $w_i \leftarrow w_j$ then $p(w_i) \leq p(w_j)$. Each release also has a planned release date, $releaseDate : [1..H] \rightarrow Date$. Many organizations use a fixed release cycle where successive release dates are separated by a constant time interval (e.g. 3 months) [9, 10, 13].

A particular case of release planning is one where the planning horizon $H = 1$. Release planning in this case is a requirements prioritization problem called "Next Release Problem (NRP)" [20, 25, 26, 27, 28, 29, 30].

2.1.3 Managing Capacity Constraints

Release planning methods must ensure that the effort required to develop the work items planned during each release period does not exceed the development team's capacity. To model this constraint, the product backlog is extended with the following information:

- $effort(w) \in \mathbb{N}^+$ denoting the estimated effort required to deliver each $w \in WI$;
- $capacity(h) \in \mathbb{N}^+$ denoting the estimated team capacity in each release period $h \in [1..H]$.

Release plans must satisfy the constraint that the effort to develop all work items in a release period does not exceed the team capacity for that period, i.e. for all $h \in [1..H]$:

$$\sum_{\{w \in WI \mid p(w)=h\}} effort(w) \leq capacity(h)$$

Effort and capacity estimates are provided by the development team and release planners. Effort and capacity must be expressed in the same unit, for example in person-days or story points. Story points are popular in lean and agile software development. They denote abstract scores on a ratio scale (e.g. a score between 0 and 100) that represent the *relative* effort needed to deliver a work item in comparison to other work items (e.g. an item worth 50 story points is expected to require 5 times the effort of an item worth 10 story points). Effort estimation in person-days denote, as the name implies, the estimated number of person-days required to deliver an item. Popular effort estimation techniques include planning poker [31], planning game [32], and formal estimation models [33] are described in the section below.

Estimating Development Effort

A recent review of effort estimation techniques [34] described existing approaches for effort estimation in software development. One such approach is Barry Boehm's constructive cost model (COCOMO) [35]. COCOMO [35] requires developers to estimate the size of features either in lines of code or function points. Effort estimates are subsequently computed through application of series of formulas on project related data. In an agile environment, team estimation game [31], planning poker [36, 23], and planning game [32] are commonly used effort estimation approaches [31, 36].

Team estimation game is a two-staged approach that allows development teams to build estimates while learning about the work before them. Team estimation game recognizes that developers find it easier to compare the complexity of features with one another even without complete information about the features. After sorting the features into various level of complexity, the team can then assign effort estimate to the features [31, 23].

In planning game, stakeholders spend a day or two-day sessions writing story cards to describe what features they want while developers assign effort estimates to those features. Planning game is based on the underlying assumption that the main stakeholders are physically available for the meeting and are able to communicate their needs and priority accurately. Stakeholders select the most promising story cards for the next release based on perceived value of the story and add it to the board until the estimated total effort matches the release capacity. Release planners use these estimates to develop an iteration plan for the release [32].

Planning poker is a consensus-based agile estimation technique [23, 36] used for estimating effort and value of software work items. In a planning poker session, each developer has a deck of planning poker cards labeled with a number from the Fibonacci sequence between 0 and 100 i.e. *0, 1, 2, 3, 5, 8, 13, ... , 100*. The product owner describes a work item to the developers and initiates discussion in order to get clarity about the work item [37, 36]. After discussion, all the developers reveal their effort estimate for the work item by choosing card. If all developers assigned the same number, that becomes the estimate of the effort for that work item. However, if the number assigned are different, the developers discuss their estimates with justifications for their choice. After the discussion, the developers select another card to re-estimate the work item based on the new information. The planning poker process is repeated until the developers reach a consensus estimate about the work item or else estimation of the work item is postponed until more information become available.

For example, Table 2.1 shows effort estimates in person-days for work items in the product backlog of our motivating example in Section 1.1. The effort required to develop work item C is estimated as 9 person-days. If the team capacity for each release is 30 person-days, then sum effort of all work items assigned to each

Work Items	Effort
A: View council tax bills	4
B: Apply for council tax reduction	9
C: Pay council tax	9
D: View council tax reduction claim	3
E: Report house move	3
F: Apply for parking permit	3
G: Buy visitor parking permit	3
H: Pay parking and traffic fine	5
I: Look up rubbish collection day	6
J: Report missed rubbish collection	9
K: Order recycling bin	7
L: Submit housing application	7
M: Report accommodation problem	5
N: Submit planning application	5
O: Comment on planning application	8
P: Create application alert	6
Q: View planning applications	7
R: Contest parking fines	3
S: Set up council tax direct debit	4

Table 2.1: Example effort estimation in person-days.

release must not exceed 30 person-days.

Constraints on other resources such as personnel, equipment, and capital can be modelled in the same way by specifying the resource requirements for each work item and the resource capacity during each period.

A well-known problem in managing capacity constraints is the difficulty of predicting development effort: effort estimates are often optimistic and ignore the inevitable uncertainty of software development tasks [38]. Ignoring such uncertainty, or at least not describing it explicitly, prevents release planners from analysing the likelihood and impacts of late deliveries. We introduce a Bayesian approach for eliciting effort uncertainty in Chapter 4.

2.1.4 Value

Software systems are built to deliver value to their stakeholders [39]. Value is defined as a measure of how much a work item w is worth to a stakeholder or group of stakeholders. The value of items in the product backlog, often derived from organization business objectives is one of the crucial factors that informs release planning decisions [40, 41]. Every work item has a value and this value is perceived differently by each stakeholder. The value of software work items can be expressed in financial terms [2, 42] or using value points [8, 25]. Release planning approaches optimizing value points are explained in Section 2.2 while approaches optimizing financial value are explained in Section 2.3.

2.2 Optimizing Value Points

To help release planners identify which release plans are likely to deliver most value, many release planning methods rely on assigning value points to work items. Value points are elicited by asking business stakeholders or product owners to provide an estimate of their perceived value of each work item. Value points estimation is often carried out using an ordinal scale e.g. a rating from 0 to 9. For example, a stakeholder might assign value point 0 to work item w if perceived to have no value and 9 if perceived to have high value relative to other work items. In a case where the project has only one stakeholder, the value point of each work item is the value assigned to the work item by that stakeholder. However, when multiple stakeholders are involved in the project, each stakeholder group is assigned a weight that denotes his/her level of importance to the business and the value point of a work item is computed as the weighted sum of the value points assigned by the different stakeholders [6, 7, 25, 27, 43]. Like story points, value points are abstract scores to compare the relative values of candidate release

plans but they do not provide concrete value measures that can be observed in the world (e.g. financial gains, number of users of online services).

The simplest approach in optimizing value points consists in extending the product backlog such that each work item w has a value point score, noted $valuePoint(w)$, denoting the relative value of w in comparison to other work items. The 'Business Value' attribute used in project management systems such as JIRA and Visual Studio Team Services are typical examples of such value points. In the next section, we briefly describe software release planning approaches that rely on value points optimization to make release planning decisions.

2.2.1 Ad Hoc Release Planning

Release planning based on ad hoc methods focuses on human intuition, capabilities and communication to decide which set of work items to be included in the next release [21]. Comparative analysis of existing release planning methods has shown that most organizations select work items informally using ad hoc methods [44, 45].

Priority Quadrant is one of the popular ad hoc release planning approach that allows release planners to identify high valued work items. Priority quadrant is a lean feature prioritization approach where the value points and effort estimates are used to plot a 2-dimensional graph. Figure 2.2 shows a value quadrant for classifying work items into its value category. The work items in the upper left quadrant are the most valuable items because they have high value points and require relatively low effort to develop. Items in the lower right quadrant are the least valuable because they have low value and require relatively high effort to implement. Items in the upper right quadrant are very valuable but require high effort to implement them. Finally, items in the lower left quadrant have low value and implementation effort. Release planners use the information obtained from

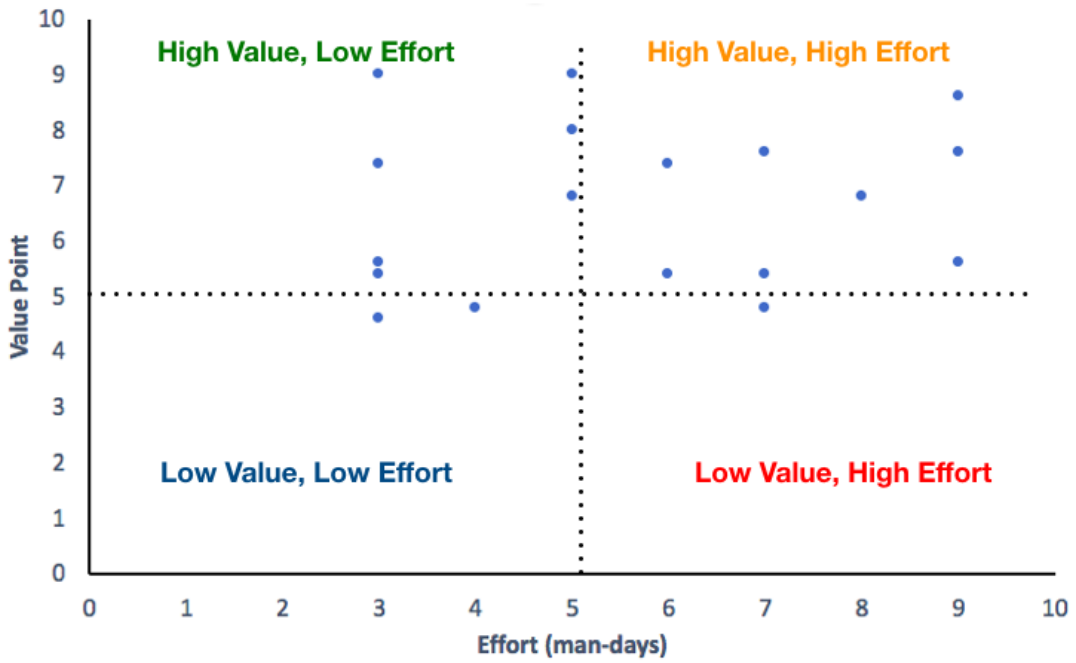


Figure 2.2: Work items value quadrant.

the priority quadrant to support decisions about what work items to include in the next release.

Ad hoc release planning methods suffer the following limitations: (i) they are not based on sound models and methodologies [8, 46, 47], (ii) they do not follow a systematic process to elicit work items dependencies and have no explicit constraint and optimization criteria [21], and (iii) they are limited to planning for just the next release [46].

2.2.2 Greedy Approaches

Greedy release planning approaches rank work items in the backlog using value point and/or effort estimates of the work items. Basic greedy approach involves ranking work items in the product backlog using their value points such that work items with more value points are ranked before those with fewer value points. Release plan is then generated by selecting items from the prioritised backlog and

assigning them to the release until capacity for the release is reached [6].

Value-effort is a greedy lean prioritization approach that helps planners in organizing and prioritizing the product backlog using story point and value point estimates. Given that $valuePoint(w)$ is the value point of work item w and $effort(w)$ is the effort required to develop w, then the value-effort ratio of w, denoted $valueRatio(w)$ is defined as:

$$valueRatio(w) = \frac{valuePoint(w)}{effort(w)} \tag{2.1}$$

Release planners can simply rank work items in the product backlog based on the value-ratio score of the items using Eq 2.1. Work items at the top of the ranked backlog can then be selected by developers for next release. For example, Table 2.2 shows the ranking of work items in the product backlog of our motivating example (see Section 1.1) based on value-effort ranking. Applying this strategy in a release means that the highest ranked work items are developed first before the lower ones until the release capacity is reached. In Table 2.2, if the team capacity for the next release is 20 person-days, then *F,D,G,N and R* will be included in the next release where F is developed first, then D and so on.

Greedy method is short-sighted because it makes decisions based on short-term value that might not be optimal in the long run while ignoring other important factors such as precedence relationship between work items, uncertainty of estimates etc. For example given three work items A, B, and C with value points 6, 7, 8 and effort estimate of 50, 30 and 100 hours respectively. Assume the capacity for the next release is 100 hours. A greedy heuristic will assign C to the next release because it has the highest value. However, an optimal solution for the next release would have been assignment of A and B with total value of 13 to the next release. We conclude that greedy release planning is not suitable for planning

Work Item	Effort	Value Point	value-effort ratio
F	3	9	3.0
D	3	7	2.33
G	3	6	2.0
N	5	9	1.8
R	3	5	1.67
E	3	5	1.67
M	5	8	1.6
H	5	7	1.4
A	4	5	1.25
I	6	7	1.16
Q	7	8	1.14
C	9	9	1.0
J	9	8	0.89
O	8	7	0.88
P	6	5	0.83
K	7	5	0.71
L	7	5	0.71
B	9	6	0.67

Table 2.2: Ranking work items by value-effort ratio

where optimality is a major criteria.

In the next few sections, we present a review of systematic release planning approaches with more sophisticated optimization techniques.

2.2.3 EVOLVE Release Planning Approaches

A number of release planning approaches using value points have been proposed over the years. Recent literature survey of release planning methods showed that 16 out of 24 reviewed approaches belong to EVOLVE family models [9]. In this section, we describe recent EVOLVE family models, variants of the model and discuss limitations of using value points in planning software releases. EVOLVE-II [6] is an improvement of the EVOLVE [21] proposed to combine capabilities of human experts and specialized optimization algorithms. EVOLVE-II models the release planning process as consisting of multiple stakeholders, multiple planning

criteria, multiple resources and different constraints. The parameters in EVOLVE-II are defined below:

- $< WI, \leftarrow >$ is the product backlog, where each work item $w \in WI$ is a candidate for the next release;
- S is the set of stakeholders whose work items are to be satisfied in the next release;
- H is the number of releases also called *release horizon*;
- C is the set of planning criteria
- $weight(c)$ is the weight of criterion $c \in C$
- $weight(s)$ is the weight of stakeholder $s \in S$;
- $weight(h)$ is the weight of release $h \leq H$;
- $valuePoint(w)$ is the value point of work item $w \in WI$;
- $effort(i)$ is the effort required to implement work item $w \in WI$;
- *capacity(h)* is the available effort for release $h \in H$;
- $score(w, s, c)$ denotes the score assigned by stakeholder s to work item w for criterion c

Where all the weighting factors and score are defined using value points.

The value point of a work item w is the weighted sum of the stakeholders assigned scores. The formula is given by:

$$valuePoint(w) = \sum_{c \in C} weight(c) \times (\sum_{s \in S} weight(s) \times score(w, s, c)) \tag{2.2}$$

In a release plan p, the value points of release h is the sum of the value points of all work items planned for that release:

$$valuePoints(p, h) = \sum_{\{w \in WI \mid p(w)=h\}} valuePoint(w) \tag{2.3}$$

The value points of release plan p is the weighted sum of the value points in each release:

$$valuePoints(p) = \sum_{h=1}^{H} weight(h) \times valuePoints(p, h) \tag{2.4}$$

where $weight(h)$ denotes the weight given to value points in release h. The release weights, to be specified by the release planners, denote the relative importance of value points delivered in earlier releases over those delivered in later releases. Formally, the weights define marginal rates of substitution between value points in different releases. For example, specifying weights 5, 3, 1 for the first three releases mean that 1 value point in the first release is equivalent to 5/3 value points in the second release and 5 points in the third release.

The objective of EVOLVE-II method is to maximize $valuePoints$(p) subject:

$$\sum_{\{w \in WI \mid p(w)=h\}} effort(w) \leq capacity(h) \tag{2.5}$$

The effort constraint guarantees that the sum effort of work items assigned to a release will not exceed the available capacity for the release. In addition to effort constraint, EVOLVE-II considers technical and precedence relationships as well. EVOLVE-II then uses an optimisation algorithm to shortlist the top 5 release plans that maximize value points (Eq. 2.4) while satisfying the capacity constraints (Eq. 2.5). Release planners can then inspect that shortlist to select their preferred release plan.

Work Item	Savings		Frequency of Use		ValuePoint
	Residents	Staff	Residents	Staff	
A	6	3	5	9	5
B	4	8	7	5	6
C	9	8	6	7	7
D	7	8	8	5	7
E	3	7	3	4	4
F	9	9	9	7	8
G	4	8	2	3	3
H	6	8	6	8	7
I	7	8	8	2	6
J	8	7	2	8	5
K	3	9	5	5	5
L	2	9	8	9	7
M	8	8	3	6	5
N	9	9	6	4	6
O	6	8	3	3	4
P	5	6	7	5	6
Q	8	7	8	6	7
R	7	3	2	5	4

Stakeholders Weights: Residents = 0.6, Staff = 0.4
Criteria Weights: Frequency = 0.3, Savings = 0.7

Table 2.3: Eliciting value points in EVOLVE II.

For example, Table 2.3 illustrates how EVOLVE-II would elicit value points for the work items in our motivating example. The two criteria of interests are 'Frequency of Use' denoting how often a service is used and 'Savings' denoting how much savings would be made by delivering this feature. The two stakeholders' groups are the local council residents and staff. Each stakeholder group is asked to score each work item against each criteria on a scale from 0 to 9. The work items' value points are then computed according to Equation 2.2. Figure 2.3 shows screenshot of release plans shortlisted by EVOLVE-II approach on our motivating example. The figure was generated using *ReleasePlanner* tool developed by the author of the EVOLVE-II approach [1].

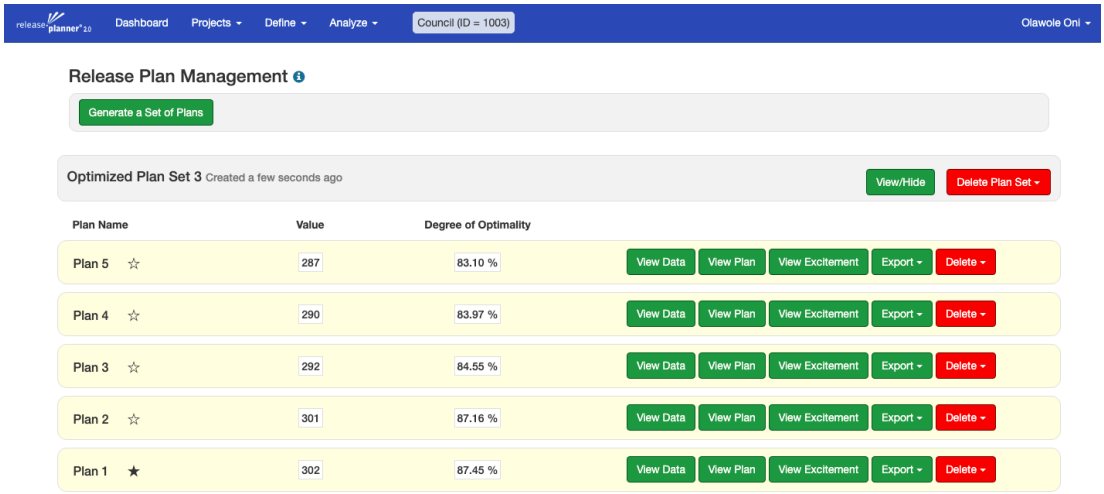


Figure 2.3: Release plan generated by ReleasePlanner web tool [1]

EVOLVE-II also supports optional post-optimization analysis such as stakeholder excitement analysis and what-if analysis [6]. Stakeholder excitement analysis consist in predicting the excitement or disappointment of individual stakeholder for a proposed plan. What-if analysis is one way of studying the impact of future scenarios on the proposed plan. What-if analysis involves measuring the impact of each input parameter on the proposed plan by changing each parameter while keeping the others constant. EVOLVE-II is supported by a web tool called ReleasePlanner [1]. EVOLVE-II has reportedly been used in industrial context for planning software releases [6].

EVOLVE-II model suffers few limitations; (i) it optimizes single objective only (ii) it lacks support for reasoning about uncertainty associated with effort and value estimates. Optimizing release plans using single objective is not sufficient because stakeholders' needs are usually conflicting with each other [25]. The conflict in stakeholders' objectives makes it difficult to combine these conflicting objectives into one single objective function because each objective measures different properties of the plan. Even though these objectives are weighted in the objective function, it is difficult to determine the weight of each objective without biasing the search to certain region of the solution space.

Ignoring uncertainty in estimation of work items effort and value can have huge consequences on the success of the project. Underestimating work items effort might result in effort overrun and missed deadline while overestimating value might result in releasing work items of low value before more valuable work items [48]. Variants of EVOLVE-II method that provide little support for reasoning about uncertainty are described in Section 2.2.5.

The following sections describe extensions of EVOLVE-II that deal with multiple objectives and with uncertainty respectively.

2.2.4 Multi-objective Release Planning Approach

Release planning decisions often involves having to take into account multiple stakeholders with conflicting objectives. Methods have been proposed to extend EVOLVE with multi-objective optimization.

Zhang et al [25] formulated next release planning as a multi-objective optimization problem called MONRP (multi-objective next release planning). MONRP is a variant of EVOLVE-II where planning horizon $H = 1$, only one planning criteria is considered and work item effort is called cost and represented using value point. Given a product backlog, set of stakeholders, and release capacity as defined in Section 2.2.3, MONRP extends EVOLVE-II by optimizing multiple objectives and define the following parameters:

- $weight(s) \in [0, 1]$ is the importance of stakeholder $s \in S$ to the organization, such that $\sum_{s \in S} weight(s) = 1$;
- $score(w, s) \in \mathbb{N}^+$ is the value point assigned to work item w by stakeholder s, such that $score(w, s) > 0$ if stakeholder s desires the delivery of work item w in the next release and zero otherwise;

The value point of a work item denoted $valuePoint(w)$ is defined as the weighted sum of the values assigned to w by the stakeholders:

$$valuePoint(w) = \sum_{s \in S} weight(s) \times score(w, s)$$

Let $\vec{x} = \{x_1, x_2, \cdots, x_n\} \in [0, 1]$ be the decision vector representing a release plan p, where decision variable $x_i = 1$ if work item w is selected in the next release and 0 otherwise [25]. The objectives of the optimization problem is to maximize value and minimize cost. The objective functions are defined as:

$$Maximize\ valuePoints(p) = \sum_{i=1}^{|WI|} valuePoint(w_i) \times x_i$$

$$Minimize\ cost = \sum_{i=1}^{|WI|} cost_i \times x_i$$

Subject to

$$\sum_{i=1}^{|WI|} cost_i.x_i \leq capacity$$

MONRP uses multi-objective meta-heuristics such as NSGA II [49], SPEA [50], and Pareto GA [51] to generate a Pareto front of solutions that optimizes the objective functions. The authors reported that NSGA II [49] algorithm perform best in term of performance and quality of solutions. Unlike EVOLVE-II, where a single release plan can dominate other plans, MONRP returns a set of Pareto optimal solutions that optimizes the conflicting objectives.

Zhang et al [26] extended MONRP to find an optimal set of work items that balances initial set of requirements to be selected against future needs. In this work, each work item is assigned an importance value called today value by stakeholders. This is based on the assumption that this value would change to a certain future value. The model therefore seeks to balance the organization's today and future

needs using multi-objective optimization. Zhang et al [52] further improves the model by proposing an approach that tackles interactions and dependencies among work items. This extension considers work items dependencies and reported the effect of the relationships on performance of various search based algorithms.

Saliu and Ruhe [53] also proposed a multi-objective decision support model called BORPES that represents release planning as a bi-objective problem that seeks to (i) maximize the business value and (ii) maximize the synergy between work items from effort saving perspectives. BORPES performs impact analysis in order to identify the elements of the existing system that will be affected by a change and assigns work items to releases based on feature coupling in the solution domain (SD-coupling) and work item business value.

Argawal et al [54] proposed a theme based release planning approach as an extension to BORPES [53]. The motivation behind theme based approach is that certain work items would have higher value if they are released along with set of inter-dependent work items. Release planning decisions in theme-based approach is made by considering trade-off between value of individual work items and synergy effects among semantically related work items. Karim et al [55] proposed further improvements by formulating theme based release planning as a bi-objective optimization problem.

The multi-objective release planning approaches discussed in this section solved the concern of single objective optimization in EVOLVE-II method. But they still lack support for reasoning about uncertainty associated with work items, delivery date, effort and value estimates. In the next section, we explain model variants that extends EVOLVE-II with uncertainty.

2.2.5 Optimizing Value Points with Uncertainty

Estimation techniques used in EVOLVE-II and variants discussed so far may result in inaccurate estimates of effort or value. Few release planning methods address this problem by estimating effort as probability distributions that express such uncertainty [15, 16, 17]. For example, some methods assume triangular probability distributions inferred from eliciting optimistic, pessimistic, and most likely efforts for each work item [15, 17]; another method assumes lognormal probability distributions [16]. Other methods evaluate release plans by computing the probability that the next release is delivered by a certain date [16, 17]. In the remaining part of this section, we briefly discuss release planning approaches that reason about uncertainty.

Ruhe et al [15] proposed EVOLVE+ to extend EVOLVE with uncertainty and risk analysis. EVOLVE+ incorporates feedback from the use of EVOLVE in the industry by modelling effort as a probability distribution and assigning risk value for each work item. EVOLVE+ uses triangular distribution to represent effort uncertainty associated with each work item. Additionally, a risk value is assigned to each work item and a risk threshold is defined to ensure that the accumulation of risk by work items assigned to a release does not exceed the threshold. Given work items' effort probability distributions, EVOLVE+ therefore evaluates release plans by optimizing weighted benefit and probability that the plan delivers all features on time [15]:

$$P(\forall h \in [1..H] : \sum_{\{w \in WI \mid p(w)=h\}} effort(w) \leq capacity(h)) \tag{2.6}$$

EVOLVE+ handles only effort uncertainty but still ignores value uncertainty which is of more importance to business stakeholders.

S-EVOLVE* [56] extends EVOLVE [21] by the(i) proactive analysis of risk involved

in adding new work items to the existing system and (ii) identifying the importance of estimating integration effort for each work items based on system characteristics. S-EVOLVE* [56] focuses on using historical defect data about work items in the existing system to estimate the risk of adding new work items. In addition to the stakeholders' satisfaction objective, this approach considered risk as an objective during the optimization. In addition to the EVOLVE-II model, S-EVOLVE* defines the following parameters:

- Let $C = \{C_1, C_2, \cdots, C_v\}$ be the set of interacting components in the existing system. Each work item w_i consist of one or more components. Hence Let $\phi(i) \subseteq C$ be the set of components in work item w_i.
- Let the health of component C_t during specific period p denoted as $H_p(C_t)$ be defined as the ratio of the number of defects affecting C_t and the total defects during the period. Period is defined here as the time between successive releases. $H_p(C_t)$ is a probability value, such that $0 < H_p(C_t) < 1$;
- Let $H(C_t)$ be the weighted average of health of component C_t over all periods.

Hence, the objective to minimize total risk likelihood of developing work item w_i in the new release was added as a new objective,

$$Minimise\ R(w_i) = 1 - \prod_{t=1}^{m} (1 - H(C_t))$$

However, the risk evaluation method used in this approach is solely dependent on the availability of historical defect and change data.

Risk Driven Method for eXtreme Programming (RDMXP_RP) [57] is a release planning method that allows developers to create a set of feasible release plan from the project profiles, analyze risk associated with each of the plan and allow

stakeholders to choose the best plan based on the result of the risk analysis. RDMXP_RP considers factors such as cost and value of work items, and work items dependencies.

REPSIM-1 [58] is a simulation model that can be used to perform risk analysis on existing release plans. REPSIM-1 uses Monte-Carlo simulation to perform sensitivity analysis on existing release plans to detect possible planning errors. REPSIM-1 was aimed to help decision makers prepare for potential risk and allow manual re-adjustment of plan for unexpected changes in requirements or decision parameters. REPSIM-2 [59] combines the risk analysis capability of REPSIM-1 with dynamic re-planning of release plans based on the outcome of the sensitivity analysis.

Fuzzy Model for Dependence constraints in Release planning (FMDCRP) [60, 61] approach builds on existing release planning methods [21, 62] by handling uncertainty using fuzzy logic. This approach models the uncertainty associated with structural dependency constraints between requirements using fuzzy logic. The satisfaction of dependency constraints in a solution plan was measured by the distance between the solution and an ideal plan. However, FMDCRP focuses only on structural uncertainty and ignores effort and value uncertainty.

Lingbo et al. [28] and Paixao et al. [29] introduced uncertainty reasoning and risk quantification to MONRP by proposing a robust optimization model that uses search based optimization and Monte Carlo simulation to find trade-off solutions with respect to risk, cost and revenue. The authors reported that the robust models are able to generate solutions that are robust and resilient under different scenarios.

Lingbo et al. [43] proposed a decision analysis framework called METRO incorporating multi-objective simulation optimization techniques, exact optimization, and uncertainty analysis to systematically aid decision support and analysis in

the presence of uncertainty. METRO framework focuses on reducing algorithmic uncertainty that originates from simulation of uncertain parameters during optimization. METRO uses the same value metric as MONRP but represents cost in financial unit unlike MONRP that uses abstract score to represent cost. Using financial metric to estimate cost can be argued to be better than abstract score but development teams in practice measure cost in term of effort required to develop a work item while financial budgets are used more as a project constraint.

2.2.6 Other Variants of EVOLVE-II methods

F-EVOLVE* [63] is a financial value-based software release planning method that extends EVOLVE [62] to accommodate financial valuation in the form of net present value. This method differs from the basic EVOLVE model in that it uses both stakeholder perceived value and actual financial value of the work items as value measures unlike EVOLVE that uses only stakeholders' perceived value. However the values of the work items are modeled as a single cash value rather than a stream of value over a specified period of time.

Albourae et al. [64] proposed a re-planning process that compares old work items to newly added work items in order to re-prioritize the work items and select the most promising ones that accommodates changes in market demands.

Post-release Analysis of Requirements Selection Quality (PARSEQ) [65] aims at finding improvement suggestions for release planning activity in a market driven software development. PARSEQ performs requirements sampling, re-estimation of cost and value, root cause analysis, elicitation of improvements and prioritization of improvements.

Regnell et al. [66] proposed an approach to extend EVOLVE using constraint solver for the optimization phase rather than evolutionary method used in EVOLVE.

Regnell et al. formulated the release planning problem as a constrained satisfaction problem and argued that using a constrained satisfaction problem (CSP) provides a more powerful, generic and flexible way of expressing prioritization with problem-specific understanding rather than focusing on algorithms. Experiments were conducted to compare the CSP approach with EVOLVE and the authors claimed that CSP is computationally stable, problem focused and more generic compared with EVOLVE. The major limitation of the CSP approach is that it requires planners to have knowledge of constrained solving and its not scalable.

Finkelstein et al [30, 67] introduced the concept of fairness in requirement analysis and optimization for analysis of trade-offs between different stakeholders' notion of fairness in requirements assignment where there are multiple stakeholders with potentially conflicting requirements preference and different views about the notion of fairness. In addition to the optimization of customer satisfaction and cost, fairness is added as an objective. The notion of fairness was represented by the number of work items fulfilled for each customer, the value and cost of the work items fulfilled for each customer. Given product backlog, value, cost and capacity as defined previously, the notions of fairness are defined as follows;

1. Fairness on absolute number of fulfilled work items: Let $NA = \{NA_1, NA_2, \cdots, NA_m\}$ denote the number of fulfilled work items for each stakeholder, where $NA_j = |WI_j|$. Hence, fairness is defined as the maximization of the average number of fulfilled work items for all stakeholders and minimization of the standard deviation of the fulfilled work items for each stakeholder.

$$Maximize\ \overline{NA}$$

$$Minimize\ \sigma(NA)$$

2. Fairness on absolute value: Let $VA = \{VA_1, VA_2, \cdots, VA_m\}$ denote the

fulfilled value for each stakeholder, where $VA_j = \sum_{i=1}^{n} value(j,i) \times x_i$. The fairness on absolute value of fulfilled work items is defined as:

$$Maximize\ \overline{VA}$$

$$Minimize\ \sigma(VA)$$

3. Fairness on the percentage of value and cost: Let $Cost\_C = \{Cost\_C_1, \cdots, Cost\_C_m\}$ denote the cost of fulfilled work items for each stakeholder and $VP = \{VP_1, VP_2, \cdots, VP_m\}$ denote the percentage of fulfilled work items value for each stakeholder, where

$$Cost\_C_j = \sum_{i=1}^{n} cost_i \times x_i,\ if\ r_i \in R_j$$

$$VP_j = \frac{VA_j}{\sum_{i:r_i \in R_j} value(j,i)} \times 100\%$$

Fairness on percentage of value and cost of fulfilled work items is obtained by optimization of these functions:

$$Minimize\ \sigma(Cost\_C)$$

$$Minimize\ \sigma(VP)$$

$$Maximize\ \overline{VP}$$

Pitangueira et al [27] extended MONRP [25] by proposing a risk-aware approach called (RA-MONRP) to address problem faced by risk-aware stakeholders when making release planning decisions. In addition to optimizing cost and average stakeholders' satisfaction objectives, RA-MONRP optimizes the dissatisfaction risk of the stakeholders. The dissatisfaction risk in RA-MONRP is a measure of the variance in the values assigned by different stakeholders to each requirement.

RA-MONRP is similar to fairness approach proposed by Finklestein et al [30] but different in the optimization approach adopted. The first phase of RA-MONRP uses MONRP to obtain the Pareto front and then allow stakeholders to select some point in the Pareto front called region of interest (ROI). These ROIs are encoded as boolean satisfiability problem and satisfiability modulo theory (SMT) solver is executed to obtain solutions around the region of interests. RA-MONRP suffers important limitation in that the approach is not scalable i.e. as the number of requirements increases, SMT solver become inapplicable.

2.2.7 Limitations of Value Point Optimization in Release Planning

Optimizing value points in software release planning is a simple and convenient way for stakeholders to prioritize work items to be included in software releases. However, release planning approaches adopting value point prioritization scheme have limitations. The following are the limitations of using value points for optimizing release plans:

1. **The validity of the elicited scores and the resulting evaluation of release plans' values should be treated with caution**. Value points are abstract scores used to compare the relative values of candidate release plans but they do not provide concrete value measures that can be observed in the world (e.g. financial gains, number of users of online services). Eliciting numerical inputs for the various weights and scores is a major difficulty and limitation of these methods. In Equation 2.2, the weights and scores correspond to marginal rates of substitutions. Eliciting such rates of substitutions is far from trivial and one can question whether the elicited numbers (such as those in Table 2.1.4) accurately reflect the stakeholders'

and release planners true preferences. With inaccurate inputs, the evaluation and ranking of release plans may also be inaccurate.

Value points cannot be directly mapped with business objectives [40, 68]. For example, we see from the Figure 2.3 in Section 2.2.3 that the quality of release plans are compared based on the value points of the plan. But the relationship between the value points and the actual objectives of the project as described in Section 1.1 is not clear. The main objective of the local council is to reduce operational cost through the usage of the web platform. How much operational cost will be saved by the council if plan with value points of 297 is selected? There is no relationship between the value points and the actual objective of the project. *Planning software releases using economic objective looks more accurate than stakeholders score on a nine-point scale but economic value are hard to get and uncertain in their nature* [6]. F-EVOLVE* [63] is the only EVOLVE model that uses financial metric but this value is treated as a one-time income rather than a stream of revenue. We introduce release planning approaches optimizing economic value in Section 2.3.

2. **They have little support for dealing with uncertainty**. The EVOLVE-II method and some of its variants assumed that business stakeholders and developers can provide accurate estimate about their perceived value and development cost of the feature respectively. However, these estimates cannot be made accurately, especially at the early stage of the project when little information is known about the project goals. Few extensions of EVOLVE-II method [15, 57, 17, 16] provide support for analysing uncertainty but most of these methods focused on effort uncertainty while ignoring value uncertainty. The difficulty for these methods to model value uncertainty originated from the fact that value point metric used is not derived from real observable project goals, therefore communicating uncertainty about these value points

is not justifiable. Few approaches that models uncertainty focuses on some type of uncertainty while ignoring others. Lingbo's [43] exact optimization method focuses on algorithmic uncertainty that originated for MC simulation and Paixao [29] method focuses more on robustness of solutions. We propose release planning approaches that uses Bayesian methods to reason about uncertainty of experts in Chapter 4.

In the next section, we'll explain release planning methods that use economic value rather than value points for optimizing software release plan.

2.3 Optimizing Economic Value in Release Planning

Even though value points allow planners to compare relative value of work items, they do not provide concrete value measures that can be observed by business stakeholders in the real world. An alternative to estimating value is to measure the value in monetary units using standard financial analysis of cost, benefits and return on investment [69, 2, 70]. Denne at al [2] presented the first release planning method that uses economic metrics for evaluating releases plans [2]. In this section, we will explain release planning methods that use economic value for value estimation and outline popular concerns resulting into low adoption of these methods.

2.3.1 The Incremental Funding Methodology (IFM)

Incremental funding method (*IFM*) [22, 2] is an economic approach to release planning in which release planning decisions are economically driven. *IFM* de-

composes the software system into units of customer valued work items known as Minimum Marketable Feature (*MMF*) and Architectural Elements (AE) which are to be developed, released and evaluated over a specific period of time. *IFM* advocates that a software development project can be optimized for financial performance by defining it in terms of *MMF*s and prioritizing the release schedule of the *MMF*s. The effect of prioritized release schedule can be observed in a project where the investment fund for financing the project is limited. A carefully prioritized release schedule can enhance the project cash flow so that revenue can be generated early in the project to offset the development costs of subsequent software features.

Figure 2.4 described various states of a software project according to IFM. The software project begins with periods of cash investment by the stakeholders. After initial periods of investment, the initial set of work items in the early release should start generating revenues and these revenues can be used to further support development of the remaining features. The project is said to achieve a self funding status when revenues from previously released work items are sufficient in supporting development of new features without additional funding from the stakeholders. Subsequently, as more valuable work items are developed and released, the project enter a repayment period during which stakeholders are refunded their initial investment capital. After the repayment period, the project should attain a break-even point. Break-even point is achieved when the investment cost is equal to total revenue generated i.e. net present value is zero. The stakeholders can then enjoy a period of profit generation by the software project afterwards.

Unlike methods described in Section 2.2, IFM does not make any consideration of resource constraints. Instead, it assumes the development team can only work on a bounded number of work items per period (e.g at most 2 work items per

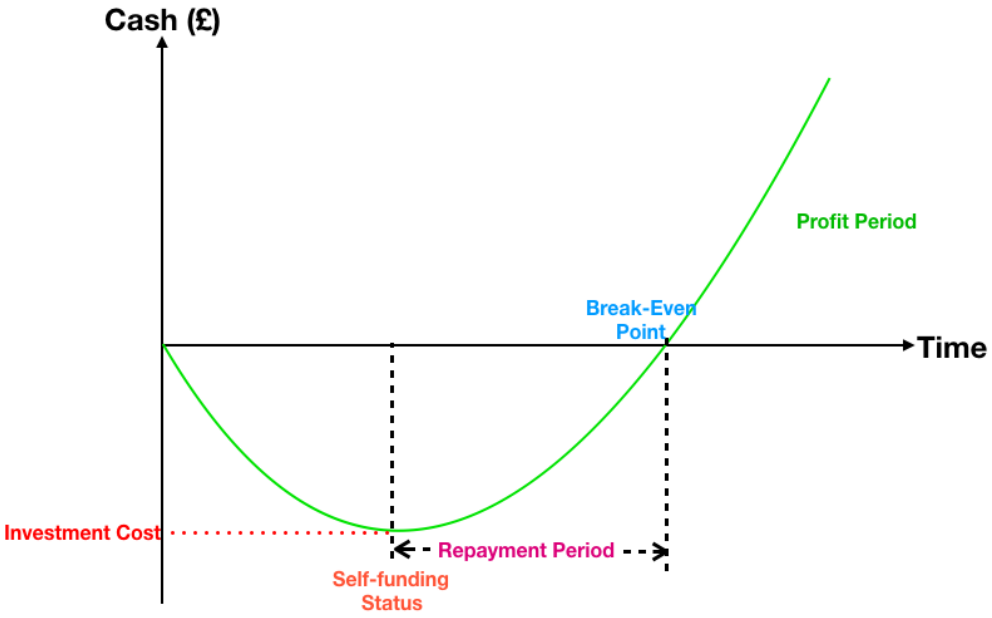


Figure 2.4: Incremental Funding Method Ideal Project [2]

period). *IFM* defined the following parameters,

- Let $WI = \{w_1, w_2, \cdots, w_n\}$ be the set of work items in the product backlog to be assigned to H subsequent releases;
- Let $cashFlow(w_i)$ be the projected cash flow of work item w_i, then $cashFlow(w_i)$ is a function $cashFlow(w_i) : [1..L] \rightarrow \mathbb{R}$ such that L is the investment horizon ($L \geq H$), i.e. the period over which the total value of the release plans will be measured, and $cashFlow(w_i, l)$ denotes the projected cost or revenue of w_i during the l^{th} period after the start of its development. For example, $cashFlow(w_i) = [-2000, 1000, 1000, 1000, 1000, 1000]$ means that developing w_i will take one period at cost of £2,000 and that once released w is projected to bring £1,000 per period for the next five periods.

In IFM, the economic value of a particular work item w_i is dependent on when it is developed. Therefore, let $value(w_i, h)$ denote the value of w_i if its development starts in release period h. $value(w_i, h)$ is defined as:

	Period											
Work Item	1	2	3	4	5	6	7	8	9	10	11	12
A	-80	10	10	12	13	14	15	15	15	15	15	15
B	-100	-50	20	20	35	40	40	40	45	50	50	50
C	-100	10	15	20	25	30	30	30	30	30	30	30
D	-150	-100	40	45	50	50	50	50	50	50	50	50
E	-100	-100	30	34	35	40	45	45	47	50	50	50
F	-100	-100	25	35	40	45	50	50	50	50	50	50
G	-120	20	24	30	36	40	45	45	45	45	45	45
H	-150	20	25	30	30	35	35	40	40	40	40	40
I	-120	20	20	25	25	30	35	35	35	35	35	35
J	-120	-100	20	30	45	50	50	55	55	60	60	60
K	-150	-150	40	50	55	60	70	75	80	80	80	80
L	-180	25	35	35	40	40	45	50	65	65	65	65
M	-200	-150	50	55	55	55	57	60	65	70	70	70
N	-100	18	20	25	25	25	25	25	25	25	25	25
O	-140	25	25	30	32	35	40	40	40	40	40	40
P	-200	15	15	18	25	35	35	35	35	35	35	35
Q	-200	-150	25	25	30	33	35	40	40	40	40	40
R	-300	30	35	35	38	40	45	45	45	45	45	45
1	-100	0	0	0	0	0	0	0	0	0	0	0
2	-150	0	0	0	0	0	0	0	0	0	0	0
3	-120	0	0	0	0	0	0	0	0	0	0	0

Table 2.4: Local Council Web Application revenue projections (in Thousand Pounds £).

$$value(w_i, h) = \sum_{l=h}^{L} \frac{cashFlow(w_i, l-h+1)}{(1+\frac{r}{100})^l} \tag{2.7}$$

where r is the discount rate and L is the investment horizon.

Net present value (NPV) is a standard economic metric for comparing the value of project investment taking into account the time value of money given by the discount rate (this allows one to take into account that £100 today are worth more than £100 in a year). Note that the discount rate plays a similar role to the release weights in the previous sections in allowing one to compare values delivered at different times. But unlike release weights, there's no need to elicit for discount rate because discount rate of an economy is publicly available. For

example, assume the discount rate per period is 2% and the development of B starts in the second period, The time value of B is computed as,

$$value(B,2) = \sum_{l=2}^{12} \frac{cashFlow(B, l+l)}{\left(1+\frac{2}{100}\right)^{l}} = 143.42$$

In the IFM method, value is more easily defined over development plans than over release plans. A development plan is a partial function $p : WI \rightarrow [1..H]$ that maps work items to the period in which their development starts. A development plan's cash flow is a function $planCashFlow(p) : [1..L] \rightarrow \mathbb{R}$ computed from the work items' projected cash flow such that $planCashFlow(p, h)$ is the sum of all work items' costs and revenues whose development started in period h. The main objective of the IFM method is to maximize the net present value. The Net Present Value (NPV) of a development plan p is then defined as:

$$NPV(p) = \sum_{h=1}^{H} planCashFlow(p, h), \tag{2.8}$$

where we have that:

$$planCashFlow(p, h) = \sum_{w_i \rightarrow h : w_i \in p} value(w_i, h) \tag{2.9}$$

IFM uses greedy heuristics with look-ahead to search for development plans that optimize NPV. The output of the IFM is rank of best development plans with highest NPV. Further analysis can be performed on these shortlisted plans to compute break-even period, investment cost, and self-funding status. Break-even period is the period in the analysis where net present value is zero. Investment cost of a plan is the maximum amount needed to fund the project. A project attains self-funding status when revenue from previously delivered features are enough to cover the cost of the remaining features to be developed. An illustration

of applying IFM to real software project is described next.

For example, to apply *IFM* to our local council example in Section 1.1. Assume that experts in the business domain estimated development cost and amount of money to be saved per period from the development of each work item as shown in Table 2.4. In the cash flow table, we assumed that each release period is 2 months and the discount rate is 12% per year. Work items *B, D, E, F, J, K, M* require two periods of development before they can be released. Table 2.5 presents the time adjusted value of work items if released in a certain period, e.g. we depict from the table that work item *A* will generate a value of £50.5K if released at the end of the first period, £38.7K if released in the second and so on.

Using a JAVA tool implementation of IFM [1], we generate release plans for serial and concurrent development release context. In a serial development context where work in progress (WIP) per period is set to one, IFM Heuristic produces an 11-period sequence $1LK.ICNAHO2$ as shown in Figure 2.5. Using equation 2.9 and 2.8, the net present value and development cost of the solution evaluates to £37.1K and £1264K respectively. For the case of concurrent development, Figure 2.6 shows the assignment of work items to releases using IFM heuristic with its corresponding net present value of £30K and development cost of £1400K. The sequence with the highest net present value is selected as the candidate release plan for the project.

2.3.2 Criticisms of Traditional IFM Approach

Even though the IFM approach is able to generate release plans that maximize the overall economic value of a software project, IFM faced a lot of criticisms especially from community of researchers that proposed value point based release

[1] https://github.com/jodal/mmfplanner

	Period											
Work Item	1	2	3	4	5	6	7	8	9	10	11	12
A	50.5	38.7	26.6	14.3	1.8	-11.0	-24.1	-36.5	-48.3	-59.4	-68.8	-78.4
B	185.7	146.3	106.1	65.1	27.4	-6.7	-41.6	-77.1	-108.8	-127.3	-146.1	-98.0
C	142.7	119.0	94.9	70.3	45.2	19.6	-6.5	-33.2	-55.8	-74.3	-88.4	-98.0
D	174.5	135.0	94.8	53.8	12.0	-30.7	-74.2	-118.6	-163.9	-205.5	-243.2	-147.1
E	170.3	130.9	90.7	49.7	10.3	-28.1	-67.3	-102.8	-134.5	-165.9	-194.2	-98.0
F	186.6	147.2	107.0	66.0	24.1	-18.5	-62.1	-102.0	-138.3	-170.6	-194.2	-98.0
G	243.9	208.4	172.2	135.3	97.6	59.2	20.0	-15.5	-48.1	-75.8	-98.4	-117.6
H	176.3	144.8	112.6	79.8	46.3	12.2	-18.3	-49.4	-76.6	-104.3	-127.8	-147.1
I	166.9	139.3	111.1	82.4	53.1	23.3	-7.2	-33.8	-56.5	-79.6	-98.4	-117.6
J	199.2	151.9	103.7	54.4	8.4	-38.5	-82.0	-126.4	-167.2	-194.9	-213.8	-117.6
K	280.7	217.6	153.3	87.6	20.7	-43.3	-104.3	-157.5	-207.3	-253.5	-291.2	-147.1
L	277.7	226.5	174.2	120.9	66.5	23.8	-15.4	-50.9	-87.1	-119.5	-152.4	-176.5
M	180.5	125.3	69.0	11.5	-42.8	-94.1	-143.7	-192.5	-242.3	-293.1	-340.3	-196.1
N	130.4	110.7	90.6	70.1	49.1	27.8	6.0	-16.2	-38.8	-61.9	-80.7	-98.0
O	197.1	165.5	133.4	100.5	67.1	32.9	-1.9	-33.0	-62.0	-89.7	-113.2	-137.3
P	77.2	48.1	20.5	-7.3	-34.4	-60.1	-86.8	-112.3	-129.0	-140.2	-149.5	-158.1
Q	-43.5	-73.8	-102.6	-131.2	-159.4	-187.9	-211.0	-232.1	-251.5	-265.7	-279.2	-290.3
R	36.6	1.4	-34.7	-68.5	-102.2	-135.8	-167.2	-194.9	-220.3	-243.4	-265.0	-272.3
1	-98.0	-96.1	-94.2	-92.4	-90.6	-88.8	-87.1	-85.3	-83.7	-82.0	-80.4	-78.8
2	-147.1	-144.2	-141.3	-138.6	-135.9	-133.2	-130.6	-128.0	-125.5	-123.1	-120.6	-118.3
3	-117.6	-115.3	-113.1	-110.9	-108.7	-106.6	-104.5	-102.4	-100.4	-98.4	-96.5	-94.6

Table 2.5: Time adjusted value of work items depending on development start period with 2% discount rate (values in Thousand Pounds £).

planning approaches. We explain the criticisms in this section and weigh-in on the validity of those criticisms. IFM approach have been criticized in the following ways:

- The proposal that IFM approach can accurately estimate future financial projection of software artifacts is unrealistic [71]. IFM approach treats software development process as a business investment and like most business investment, it relies on the ability of business experts to accurately estimate financial projection of software features. However, software development is a more dynamic process compared to other economic investment and will be unrealistic to accurately estimate the future values of software features with certainty. There are many unknown factors that can cause variations in estimated financial projection of software artifacts. This type of uncertainty is usually referred to as epistemic uncertainty [72]. Epistemic uncertainty arises when the value of a decision parameter is not accurately measured or some effects affecting the parameter are deliberately ignored [72]. Ignoring

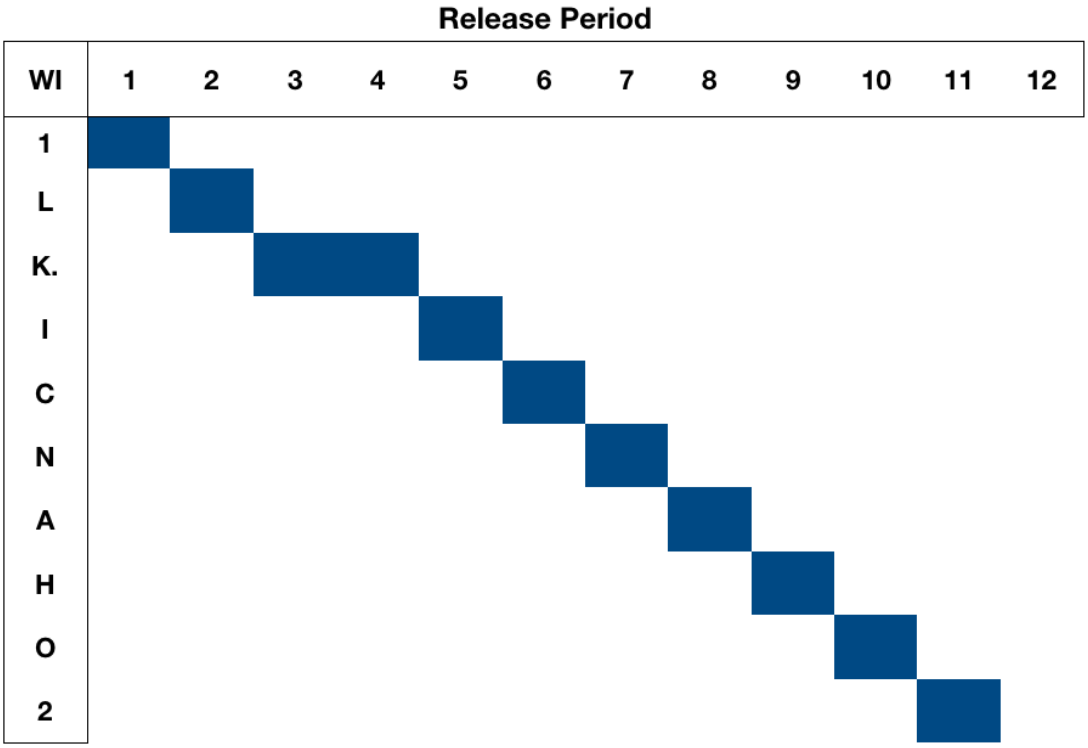


Figure 2.5: Optimal solution produced by IFM heuristics for serial development.

epistemic uncertainty in the IFM approach can lead to over-estimation or under-estimation of cost and value of software features. This criticism of IFM approach is reasonable and completely valid.

- It is impossible to assign financial value to intangibles. Using value point as a measure of value is common in release planning research community because researchers believe that the value of some software features cannot be financially quantified. These values are known as intangible values. According to [68, 73], anything can be measured directly or indirectly if there's an understanding of what's being measured and ways to measure it. IFM measures intangibles through pair-wise comparison of intangible features with tangible features. The theory adopted by IFM is that even if an amount cannot be assigned to an intangible feature directly, an estimate of such feature can be derived when the feature is pair-wisely compared to other features with known financial value. For example, assume the value of

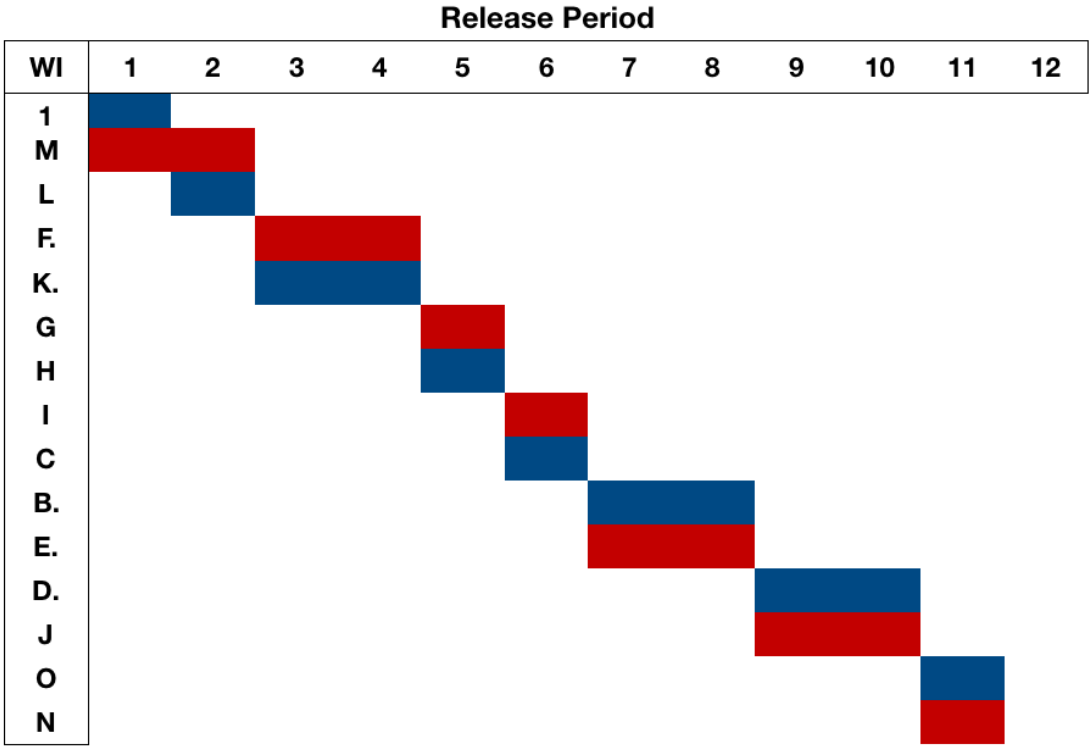


Figure 2.6: Optimal solution produced by IFM heuristics for concurrent development.

feature A is known to be $£10K$ and feature B and C are intangible features. If we know that B is twice as valuable as A and C is 1.25 times as valuable as A. Then we can infer that the values of B and C are $£20K$ and $£12.5K$ respectively. Sometimes, quantifying intangibles is not as straight forward as the example given above but more often than not, intangibles can be quantified if the value drivers for that intangible can be identified. Hubbard [68, 73] argues that anything can be measured if enough information is gathered to reduce uncertainty about the parameter of interest.

- Assumptions in IFM approach are unrealistic. IFM assumes that a fixed number of features should be developed in each iteration and the development of those features are completed in the iteration specified in the plan. This assumption is rigid and not realistic in practice. In practice, it's common for software organizations to have a fixed date, flexible scope style of release

where release cycles are defined and developers build as many features as possible within the resource limit in-between releases.

- Economic value is not the only concern of stakeholders. IFM approach only focuses on maximizing financial value of the software project. While money is important, there are other objectives that are not directly related to financial value such as minimizing investment cost, maximizing customer satisfaction, maximizing utility etc. It would be better if the approach is able to generate more release plans using more than one objective.

Extensions to the original IFM method include improved algorithms for identifying optimal development sequences [71], analysis of cash flows uncertainty [18], and analysis of competitors' behaviours using game theory [74].

Barbosa et al [18] presents a method for identification of investment policy that allows managers to make better decisions during execution of a software project with managerial flexibility. The method permits project managers to device a policy that maximizes the number of *MMFs* built during the life cycle of the project. Such investment policy is derived from a time dependent classification tree expressed as a decision tree and makes it easier to understand the decisions needed to be made as the project progresses. In their work, they proposed that having an investment policy brings about an increase in project value while maintaining the risk of financial loss under an acceptable threshold [18].

Cantor [70] developed a method based on an assumption that determining the value of a development program requires dealing with its future cash flow and requires computing NPV of uncertain costs and benefits. The uncertain cash flow was modelled using a triangular distribution. Unlike *IFM*, this method treats the entire project as a single *MMF* and attempts to determine how soon the project can deliver value to the business.

A Statistical Approach [71] to *IFM* proposed an approximation method to solve a problem with a large set of interconnected *MMF*s and AEs where it is infeasible to consider all possible delivery sequences. This approach provides a statistical approximation of solutions with an arbitrary degree of confidence. Alencar et al. [71] suggested that using the statistical approach can make developers and project managers feel more confident about the quality of the decisions they made. It was an improvement to *IFM* because it yields a dependable estimate for approximation error that might have occurred during the Monte Carlo simulation of uncertain cash flows.

Eduardo et al [75] proposed an approach for the maximization of software projects financial returns under duopolistic market situation based on the application of Game Theory concept to *IFM*. Their work identified one of the weaknesses of *IFM* that it ignores the effects of competition that exist in the business context. *IFM* process was extended to handle competition by modelling *IFM* as a strategic game between two players and obtain results using games solution technique known as Nash Equilibrium.

Majority of the proposed extensions of the IFM approach has focused mainly on introducing uncertainty to cash flow projections but they still rely on same assumptions as the traditional approach and optimizes a single objective. The IFM extensions for analyzing cash flow uncertainty assume a single work item can be worked on at a time and flexible release dates. They are thus not suitable for the more complex problem of analyzing uncertain cash flows with fixed release dates. In Chapter 3, we investigate the feasibility of extending IFM with uncertainty and multi-objective optimization.

2.4 Summary and Conclusion

This chapter reviewed state of the art release planning methods. We explained basic concepts of software release planning and set of activities involved in release planning. We classified release planning approaches into methods that optimize value points and methods that optimizes financial value. Table 2.6 presents a summary of the different release planning methods covered in this chapter. The table classifies existing release planning approaches based on eight different criteria. The criteria are number of releases, architecture decision, value metrics, dependency relationship, tool support, uncertainty, and multi-objective optimization.

Number of releases indicates the number of releases supported by the release planning method. Architecture decision describes if architecture decisions are considered in the approach during the generation of release plan solutions. Architecture decisions has great influence on the value, cost and quality attributes of the system. IFM based approaches considers the impact of architecture decisions on release plan solutions while the other approaches assumed that architecture elements are composed within the features or requirements. Value metrics denotes whether the release planning method uses value point or economic valuation. Dependency relationship signifies if the approach consider dependency relationships between work items. Tool support signifies if the release planning method is supported by a publicly available software tool. Uncertainty signifies whether the release planning method considers any form of uncertainty modeling. Finally, multi-objective criteria signifies whether single or multi-objective optimization is adopted by the release planning methods.

Methods optimizing value points use information about stakeholders' preferences represented as value points, release weights, planning criteria, effort and dependency to generate plans for upcoming releases of the software. We studied family

of EVOLVE models and their variants. Limitations identified with optimizing release plans using value points are; (i) use of value points as value metric makes it easy to prioritize work items but these value points does not translate to business goals that can be observed in the real world (ii) they provide little or no support for analysing uncertainty. Few variants of EVOLVE method that consider uncertainty in their model formulation deals with either effort uncertainty, structural uncertainty or algorithmic uncertainty but largely ignores value uncertainty. Incremental funding methodology (IFM) introduced an economic approach to release planning where release planning decisions are financially driven. We identified popular criticisms of IFM approach in the release planning community and IFM extensions proposed to respond to those criticisms. The following conclusions can be made from the models reviewed in this chapter:

1. NRP, EVOLVE and IFM models have unique strengths and weaknesses as explained earlier. There's no approach that has been to combine the strengths of these model families while eradicating most of their weaknesses.

2. Existing approaches lacks support for analysing value uncertainty

3. Existing models are not suitable for release planning where release dates are fixed and release scope is flexible

4. There's no empirical study to investigate the effect of reasoning about uncertainty in release planning.

In Chapter 3, we propose an extension to IFM method to support uncertainty and multi-objective optimization. In Chapter 4, we propose a Bayesian framework for software release planning under different release scenarios that mitigated limitations described above.

Release Planning Method	No of Releases	Arch Decision?	Value Metric	Uncertainty/ Risk	Multi-Objective?	Tool Support?	Dependency Relationship?
EVOLVE [21]	2	No	Value Point	No	No	Yes	Yes
EVOLVE* [62]	2	No	Value Point	No	Yes, weighted	Yes	Yes
EVOLVE+ [15]	2	No	Value Point	Yes	Yes, weighted	Yes	Yes
S-EVOLVE* [56]	3	No	Value Point	Yes	Yes, weighted	Yes	Yes
F-EVOLVE* [63]	1	No	Value Point & Monetary	Yes	Yes	Yes	Yes
Lightweight replanning [64]	1	No	Value Point	No	Yes	No	Yes
MONRP [25]	1	No	Value Point	No	Yes	No	No
Fairness Analysis [30]	1	No	Value Point	No	Yes	No	No
RA-MONRP [27]	1	No	Value Point	Yes	Yes	No	No
REPSIM-1 [58]	1	No	Value Point	Yes	No	Yes	Yes
REPSIM-2 [59]	1	No	Value Point	Yes	No	Yes	Yes
RDMXP_RP [57]	1	No	Value Point	Yes	Yes	Yes	Yes
BOPRES [53]	1	No	Value Point	No	Yes	No	Yes
FMDCRP [60]	1	No	Value Point	No	No	No	Yes
Robust NRP [28]	1	No	Value Point	Yes	Yes	No	Yes
PARSEQ [65]	1	No	Value Point	No	No	Yes	Yes
IFM [2]	Multiple	Partially	Monetary	No	No	Yes	Yes
Investment Policy [18]	Multiple	Partially	Monetary	Yes	No	No	Yes
Improving ROI [70]	1	No	Monetary	Yes	No	No	No
Statistical IFM approach [71]	Multiple	Partially	Monetary	Yes	No	No	Yes
Game Theory IFM [75]	Multiple	Partially	Monetary	No	No	No	No

Table 2.6: Summary of Release planning models and factors treated by the models

Chapter 3

Cost-Value Based Release Planning with Uncertainty

We concluded from Chapter 2 that state of the art release planning methods do not provide adequate support reasoning about uncertainties associated with business value and effort. The objective of the work described in this chapter is to propose an approach called *MOIFM* that extends standard *IFM* with reasoning about uncertainty and multi-objective optimization.

The elicitation of work items and dependency constraints follow the same process as the standard *IFM*. However, our approach differs from *IFM* in that business cash flow values will be represented as stochastic values instead of deterministic estimates and release plans are optimized using multiple stakeholders objectives i.e. net present value, investment cost and investment risk. The method presented here improves standard *IFM* but still has important limitations: it assumes that single work item can be worked on at a time and it is not scalable due to its use of an exhaustive search technique. Most importantly, the method is not able to support the industrial practice of release cycles with fixed dates and flexible scopes. All these limitations will be addressed in Chapter 4.

3.1 Extending IFM with Uncertainty

Estimating cash flow of work items is a non-trivial task that requires meeting with several stakeholders and domain experts in the business context to get their subjective opinion about business value and development cost. Technical experts are skilled in estimating the cost of developing work items while business experts are able to estimate revenue generating potential of work items. These estimates are provided with some degree of uncertainty in the mind of the estimators due to incomplete information at the time the estimate was given. Uncertainty arises from the lack of perfect information about the software project, especially at the early phases of development process when work items estimation is done. Standard *IFM* approach lacks support for helping domain experts in analysing uncertainty about cost and value of work items. To incorporate uncertainty into *IFM* approach, we propose two ways of eliciting uncertainties for work items cash flows.

3.1.1 Eliciting Uncertainty as Triangular Distribution

Eliciting triangular probability distribution requires that cost and value of work items be elicited from stakeholders through a three-point estimation method. Instead of eliciting for a deterministic point estimate for the value or cost of a work item, the release planner elicits the plausible minimum (a), mode (c), and maximum (b) values of the work item. The minimum value represents the lowest possible value, maximum value represents the highest possible value while the mode represents the most likely value of the quantity being estimated where $a \leq c \leq b$. *MOIFM* interprets these three estimates as a triangular probability distribution $triangle(a, c, b)$ as represented in Figure 3.1 from which random samples X can be drawn with associated probability $Pr(X)$. We observe from the

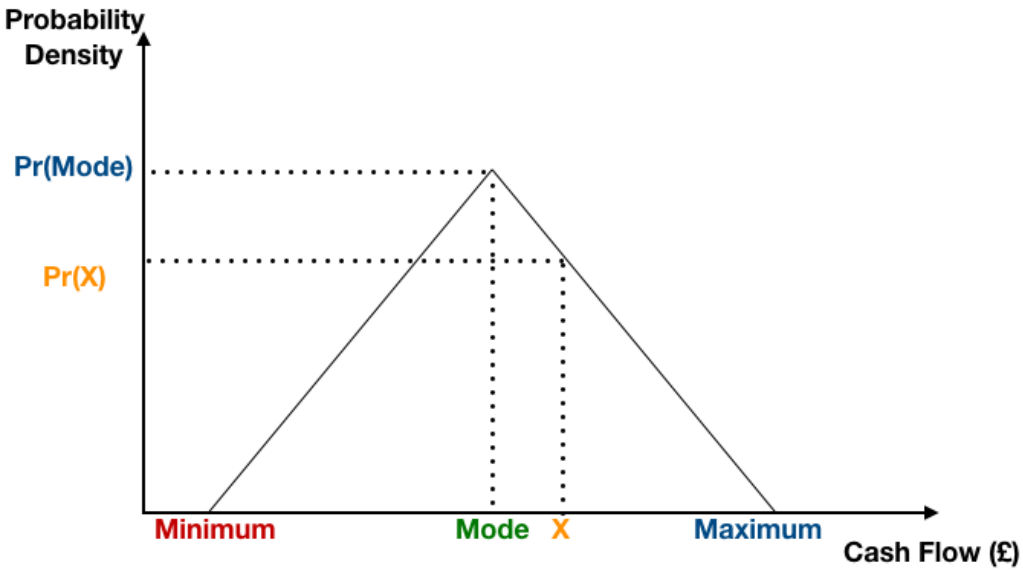


Figure 3.1: Triangular Distribution.

figure that the mode represents the value with the highest likelihood of occurrence.

For example, instead of the point estimate in Table 2.4, suppose we want to elicit the cost of developing work item A. An expert might think that the cost of implementing work item A is at least $70K$, at most $120K$ and most likely to be $90K$. *MOFIM* represents such estimate as a triangular distribution $triangle(70, 90, 120)$.

Estimating probability distribution of cost and value for work items that have not been developed is difficult. The estimation process requires domain experts that possess wealth of knowledge and experience in the application domain and business context. However, there are statistical methods that have been applied in other engineering fields to train experts and facilitate such elicitation process [72, 76].

3.1.2 Adding Uncertainty to Point-Based Estimates

Today, cost and value estimates are commonly given as point-based estimates. But these point based estimates are sometimes too optimistic or pessimistic [48]. In

this section, we propose a simple method for transforming point-based estimates into triangular distribution using past project data about deviations between predicted and actual values.

Recent findings has shown that 90% of software projects tend to underestimate cost and overestimate revenue [48, 77]. This is because project managers tend to present a low cost estimates and high revenue generating proposal to financial stakeholders in order to get project investment approval. Standish CHAOS report [78, 79] also reported that 52.7% of software projects experienced cost overrun by at least 80%. Based on these findings, we assume that there is a slim chance that a work item will cost less than the projected amount but will likely cost about 80% more than the estimated amount. Similarly, a work item that is estimated to generate a certain revenue over some period of time might not generate up to the estimated amount, might generate more, or might not generate any value within the period of time under evaluation.

To add uncertainty to the point-based cost estimate, we define actual cost as the sum of the point-based estimate plus the uncertain cost overrun. Hence we have that:

$$actualCost = pointCE + costUncert \tag{3.1}$$

Where *pointCE* is the deterministic cost estimate assigned to a work item and *costUncert* is a random variable:

$$costUncert = triangle(minCO, modeCO, maxCO) \tag{3.2}$$

Where *minCO*, *modeCO* and *maxCO* are expected minimum, mode and maximum cost overrun respectively.

Similarly, we define the actual value to be generated by each work item as a random variable,

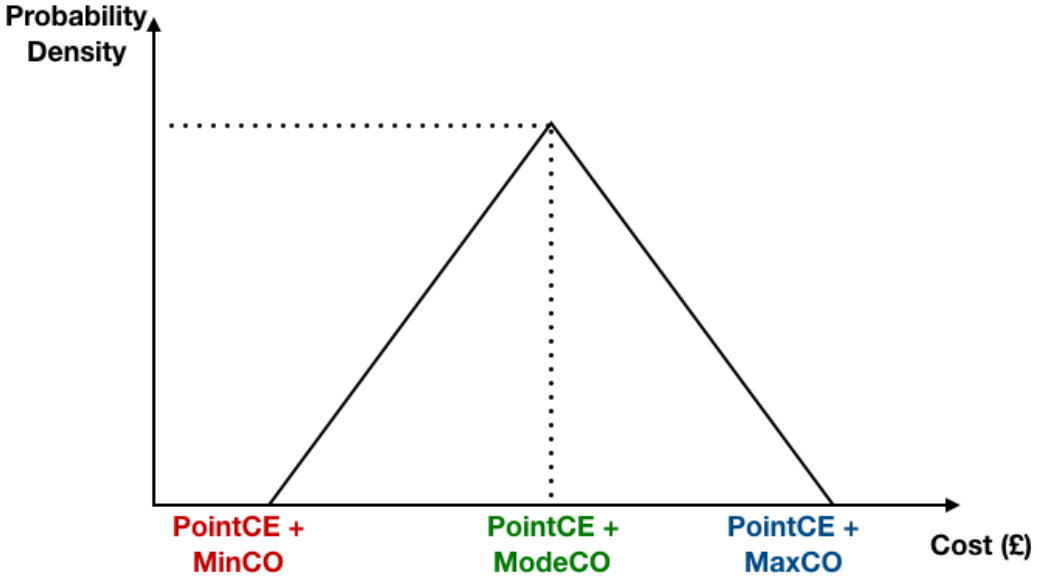


Figure 3.2: Deriving cost distribution from point estimate cash flows

$$actualValue = triangle(0, pointVE, (pointVE + valueError)) \tag{3.3}$$

Where *pointVE* is the point-based estimate of value assigned to the work item in a given period and *valueError* is a quantity denoting estimation error for work item value, *valueError* is zero if value is perfectly estimated. Figure 3.2 and 3.3 showed the derivation of cost and value of work items from point estimates.

For example, to add uncertainty to the cash flow estimates in Table 2.4, if we assume that the project will most likely incur a cost overrun of 20% of the deterministic estimate and cost overrun cannot be more than 45% of the predicted cost. Hence, we defined the cost distribution *costUncert*,

$$costUncert = triangle(0, 0.20 * pointCE, 0.45 * pointCE) \tag{3.4}$$

Where the mode is 20% of *pointCE*, minimum and maximum cost overrun are 0% and 45% of pointCE respectively. Similarly, if we assume a 20% error margin for the estimated value, then the distribution of the value generated by each work

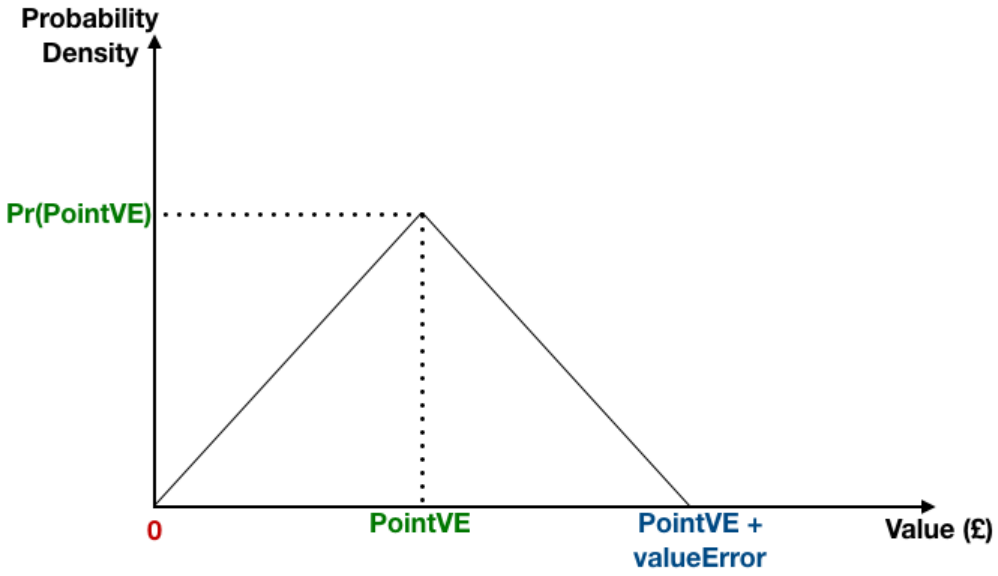


Figure 3.3: Deriving value distribution from point estimate cash flows.

item is denoted as:

$$value = triangle(0,\ pointVE,\ 1.2 * pointVE) \tag{3.5}$$

Where $pointVE$ is the mode, 0 and $1.2 * pointVE$ are the minimum and maximum value respectively. Table 3.1 shows the resulting probability distribution of the cash flow elements after applying equations 3.4 and 3.5 for computing cost and value distributions respectively.

3.1.3 Simulation Cash Flow Projections

We use Monte Carlo Simulation to generate N cash flow scenarios by randomly sampling from the probability distributions of the work items cash flows. The outcome of the simulation process is a $N \times m \times l$ three-dimensional array, where N is the number of simulations, m is the number of work items and l is the investment horizon. In this way, rather than having just one cash flow projection

Work Item	Period 1	2	3	4	5	6	7	8	9	10	11	12
A	-(80,96,116)	(0,10,12)	(0,10,12)	(0,12,14)	(0,13,15)	(0,14,16)	(0,15,18)	(0,15,18)	(0,15,18)	(0,15,18)	(0,15,18)	(0,15,18)
B	-(100,120,145)	-(50,60,72)	(0,20,24)	(0,20,24)	(0,35,42)	(0,40,48)	(0,40,48)	(0,40,48)	(0,45,54)	(0,50,60)	(0,50,60)	(0,50,60)
C	-(100,120,145)	(0,10,12)	(0,15,18)	(0,20,24)	(0,25,30)	(0,30,36)	(0,30,36)	(0,30,36)	(0,30,36)	(0,30,36)	(0,30,36)	(0,30,36)
D	-(150,180,217)	-(100,120,145)	(0,40,48)	(0,45,54)	(0,50,60)	(0,50,60)	(0,50,60)	(0,50,60)	(0,50,60)	(0,50,60)	(0,50,60)	(0,50,60)
E	-(100,120,145)	-(100,120,145)	(0,30,36)	(0,34,41)	(0,35,42)	(0,40,48)	(0,45,54)	(0,45,54)	(0,47,56)	(0,50,60)	(0,50,60)	(0,50,60)
F	-(100,120,145)	-(100,120,145)	(0,25,30)	(0,35,42)	(0,40,48)	(0,45,54)	(0,50,60)	(0,50,60)	(0,50,60)	(0,50,60)	(0,50,60)	(0,50,60)
G	-(120,144,174)	(0,20,24)	(0,24,30)	(0,30,36)	(0,36,43)	(0,40,48)	(0,45,54)	(0,45,54)	(0,45,54)	(0,45,54)	(0,45,54)	(0,45,54)
H	-(150,180,217)	(0,20,24)	(0,25,30)	(0,30,36)	(0,30,36)	(0,35,42)	(0,35,42)	(0,40,48)	(0,40,48)	(0,40,48)	(0,40,48)	(0,40,48)
I	-(120,144,174)	(0,20,24)	(0,20,24)	(0,25,30)	(0,25,30)	(0,30,36)	(0,35,42)	(0,35,42)	(0,35,42)	(0,35,42)	(0,35,42)	(0,35,42)
J	-(120,144,174)	-(100,120,145)	(0,20,24)	(0,30,36)	(0,45,54)	(0,50,60)	(0,50,60)	(0,55,66)	(0,55,66)	(0,60,72)	(0,60,72)	(0,60,72)
K	-(150,180,217)	-(150,180,217)	(0,40,48)	(0,55,66)	(0,55,66)	(0,60,72)	(0,70,84)	(0,75,90)	(0,80,96)	(0,80,96)	(0,80,96)	(0,80,96)
L	-(180,216,261)	(0,25,30)	(0,35,42)	(0,35,42)	(0,40,48)	(0,40,48)	(0,45,54)	(0,50,60)	(0,65,78)	(0,65,78)	(0,65,78)	(0,65,78)
M	-(200,240,290)	-(150,180,217)	(0,50,60)	(0,55,66)	(0,55,66)	(0,55,66)	(0,57,69)	(0,60,72)	(0,65,78)	(0,70,84)	(0,70,84)	(0,70,84)
N	-(100,120,145)	(0,18,21)	(0,20,24)	(0,25,30)	(0,25,30)	(0,25,30)	(0,25,30)	(0,25,30)	(0,25,30)	(0,25,30)	(0,25,30)	(0,25,30)
O	-(140,168,203)	(0,25,30)	(0,25,30)	(0,30,36)	(0,32,38)	(0,35,42)	(0,40,48)	(0,40,48)	(0,40,48)	(0,40,48)	(0,40,48)	(0,40,48)
P	-(200,240,290)	(0,15,18)	(0,15,18)	(0,18,21)	(0,25,30)	(0,35,42)	(0,35,42)	(0,35,42)	(0,35,42)	(0,35,42)	(0,35,42)	(0,35,42)
Q	-(200,240,290)	-(150,180,217)	(0,25,30)	(0,25,30)	(0,30,36)	(0,33,39)	(0,35,42)	(0,40,48)	(0,40,48)	(0,40,48)	(0,40,48)	(0,40,48)
R	-(300,360,435)	(0,30,36)	(0,35,42)	(0,35,42)	(0,38,45)	(0,40,48)	(0,45,54)	(0,45,54)	(0,45,54)	(0,45,54)	(0,45,54)	(0,45,54)
1	-(100,120,145)	0	0	0	0	0	0	0	0	0	0	0
2	-(150,180,217)	0	0	0	0	0	0	0	0	0	0	0
3	-(120,144,174)	0	0	0	0	0	0	0	0	0	0	0

Table 3.1: Deriving triangular distributions from point estimate.

as in the case of the standard *IFM*, we have N different cash flow projections for each work item.

3.2 Multi-objective Optimization Extension

In addition to the net present value objective used in the traditional IFM approach, we introduced two objectives and optimize release plans using three objectives. In our approach, release plans are optimized using Expected Net Present Value ($ENPV$), Expected Investment Cost (EIC), and Investment Risk (IR). Given a release plan p, the objectives are explained below:

3.2.1 Expected Net Present Value

Expected Net Present Value ($ENPV$) is the average net present value over all cash flow scenarios. Rather than computing net present value of release plans using point-based cash flow projections as in the case of traditional *IFM* approach, *MOIFM* computes expected net present value over multiple scenarios of cash flow projections. For each scenario of cash flow generated during simulation, *MOIFM*

uses equation 2.8 to compute the net present value of the plan for that scenario. $ENPV$ is defined as:

$$ENPV(p) = E[NPV(p)] \tag{3.6}$$

3.2.2 Expected Investment Cost

Investment cost of a release plan p is the total discounted amount required to fund the plan. We illustrated investment cost in Figure 2.4 as the point when no additional investment is required from the stakeholders to complete the project. Let $IC_h(w)$ be the cost of developing work item w given that its development starts in period h, then the investment cost of plan p denoted $IC(p)$ is defined as:

$$IC(p) = \sum_{w \to h : w \in p} IC_h(w) \tag{3.7}$$

The investment cost is computed for each cash flow scenario during the simulation. The expected investment cost of a release plan denoted EIC(p) is therefore defined as the mean investment cost over all scenarios of cash projections.

$$EIC(p) = E[IC(p)] \tag{3.8}$$

3.2.3 Investment Risk

The investment risk of a release plan denoted IR(p) is defined as the coefficient of variation of the net present value [70]. It is calculated as the ratio of the standard deviation of the net present value and the expected net present value. Investment risk of a plan measures the risk of the software project generating the expected

net present value if the plan is implemented.

$$IR(p) = \frac{\sigma[NPV(p)]}{E[NPV(p)]} \tag{3.9}$$

3.3 Optimizing Release Plans

We use exhaustive search to generate all release plans of the software project that satisfies the precedence relationships. In the worst case scenario when work items has no dependency, the number of release plans generated will be $n!$ where n is the number of work items in the project. For each valid release plans generated, We compute the expected net present value, expected investment cost and investment risk using equations 3.6, 3.8, and 3.9 respectively.

For example, if we were to generate all release plans in our motivating example, we'll need to generate $6.4E+15$ release and then filter the plans using precedence rule to eliminate plans that violates precedence constraints. To simplify the example, assume we only consider the first 10 work items in the motivation example. We generate $3,628,800$ release plans and after applying the precedence rule, we have $432,212$ valid release plans.

MOIFM computes the shortlist as the set of pareto optimal release plans for the objective values (*ENPV*, *EIC* and *IR*). Our shortlist approach is an extension of the standard Pareto optimality method explained in Section 2.2.4. Due to statistical error in the Monte Carlo Simulation, shortlisting candidate release plans based on strict Pareto optimality can result in exclusion of valid release plans due to a little difference in one of the objective values. We neutralized the error by specifying resolution margins for decision criteria to ensure that shortlisted release plans are clearly better than other alternatives by a significant margin. For example, if the resolution margin for the *ENPV* is set to £5K, then comparing

the *ENPV* of release plans will ignore differences between ENPVs less than £5K.

Figure 3.4 shows the Pareto front for the motivation example. Though the releases were generated using three objectives, we use a 2D visualization for simplified view. The cross points represent the shortlisted release plans while the circle dots represent the dominated solutions. The figure shows the trade-off between maximizing net present value and minimizing investment cost. We see from the figure that release plans that generates high *ENPV* are likely to require high investment cost and vice versa. Decision makers can then analyse the shortlisted release plans before making decision on what plan to choose.

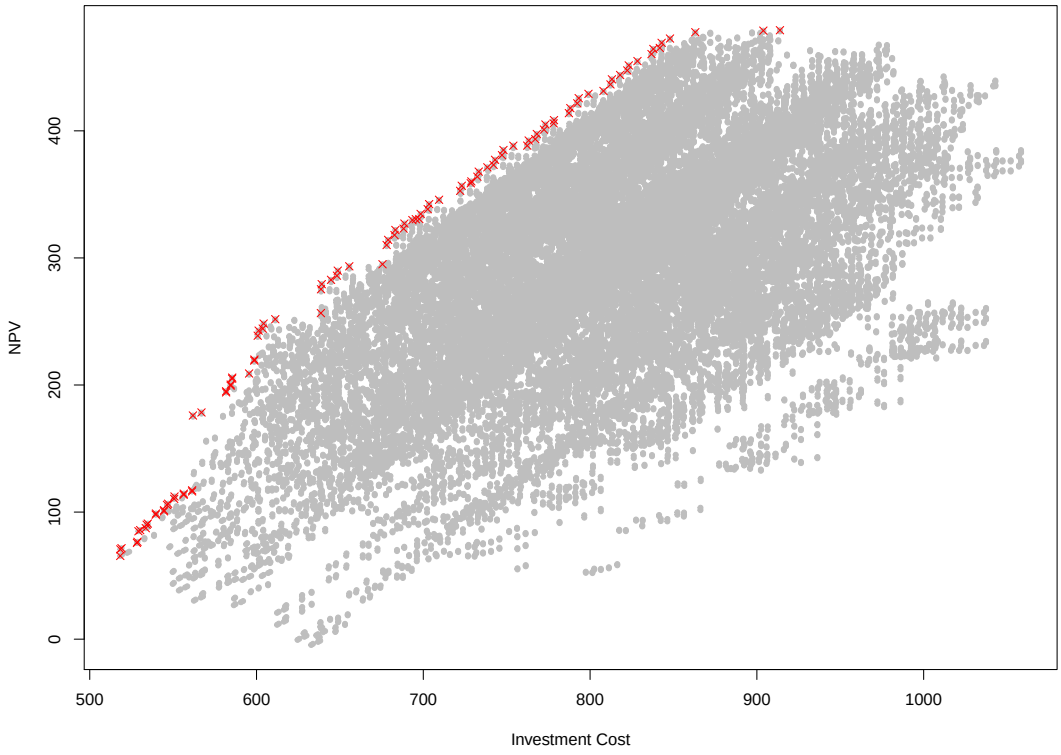


Figure 3.4: Shortlisted release plans for Local Government Project.

3.4 Summary, Conclusion and Limitations

In this work, we propose an approach to test the feasibility of extending *IFM* to deal with uncertainty and multiple objectives. To achieve this, we developed a prototype tool (in R programming language) and applied it to the motivation example introduced in Section 1.1. Our tool has the following capabilities:

1. Represents uncertainty about work items cash flows as triangular distributions. A triangular distribution is characterized by three parameters specifying the lowest, most likely, and highest value for a variable. We chose this distribution because it is easily understood and used in IT portfolio management tools [70]. We envision, however, extending our tool to additional probability distributions based on empirical research on software cost and value estimations and methods to elicit uncertainty from experts [72].

2. Uses Monte Carlo simulation to compute the impact of work cash flow uncertainty on the *NPV* of alternative release plans. For each release, our tool computes expected *NPV* (the mean *NPV* over all simulations), expected investment cost (the mean of the total cost to be invested in the project before it has a positive cash flow), and its investment risk (the ratio between its *NPV* standard deviation and its expected *NPV* [70]).

3. The statistics about the *NPV* simulations are then used to select the Pareto-optimal set of release plans that maximize expected *NPV*, minimize expected investment cost, and minimize investment risks. We have chosen these objectives because they are used in IT project portfolio management tools [70]. Decision makers can, however, select alternative set of optimization objectives that suits their context.

The implementation of multi-objective *IFM* has the following limitations:

1. Like the standard *IFM* algorithm, it assumes the number of work items that can be developed during each release period is fixed and development time for each work is one release period. This is not always the case in real life project, some work items might be larger and require more time to develop and will likely span more than one development period. There's also the likelihood that the development team fails to deliver a work item at the appropriate time which will negatively impact the projected value of the software project. In practice, software projects allows concurrent development of work items in which different developers work on different work items simultaneously.

2. It uses an exhaustive search to identify Pareto-optimal release plans which limits its scalability to problems involving no more than a dozen work items. The proposed approach in this chapter does not scale well. The run-time increases exponentially as the number of work item increases.

In the next chapter, we propose an approach that deals with the limitations above by combining the strengths of *IFM* and *EVOLVE* approaches.

Chapter 4

Release Planning with BEARS

In chapter 3, we investigated the feasibility of extending IFM with uncertainty and multi-objective optimization. The proposed extension represented cash flow uncertainty as a triangular distribution and introduced two new objectives to the traditional IFM; but still suffers limitations inherited from the traditional IFM. These limitations are outlined and explained in Section 3.4. The limitations of release planning methods explained in Chapter 2 and Section 3.4 make them unsuitable for supporting software release planning decisions in a release process where release date is fixed, work scope is flexible, effort required to develop features are uncertain and value of software features are uncertain. In this chapter, we propose a *Bayesian Economic Analysis of Release Scenarios* (BEARS) framework to support software release planning under uncertainty. BEARS framework aims to help decision makers with making release decisions that are driven by measurable business objectives rather than abstract objectives in the presence of uncertainty. BEARS approach is suitable for release planning process where release dates are fixed and work items scope are flexible.

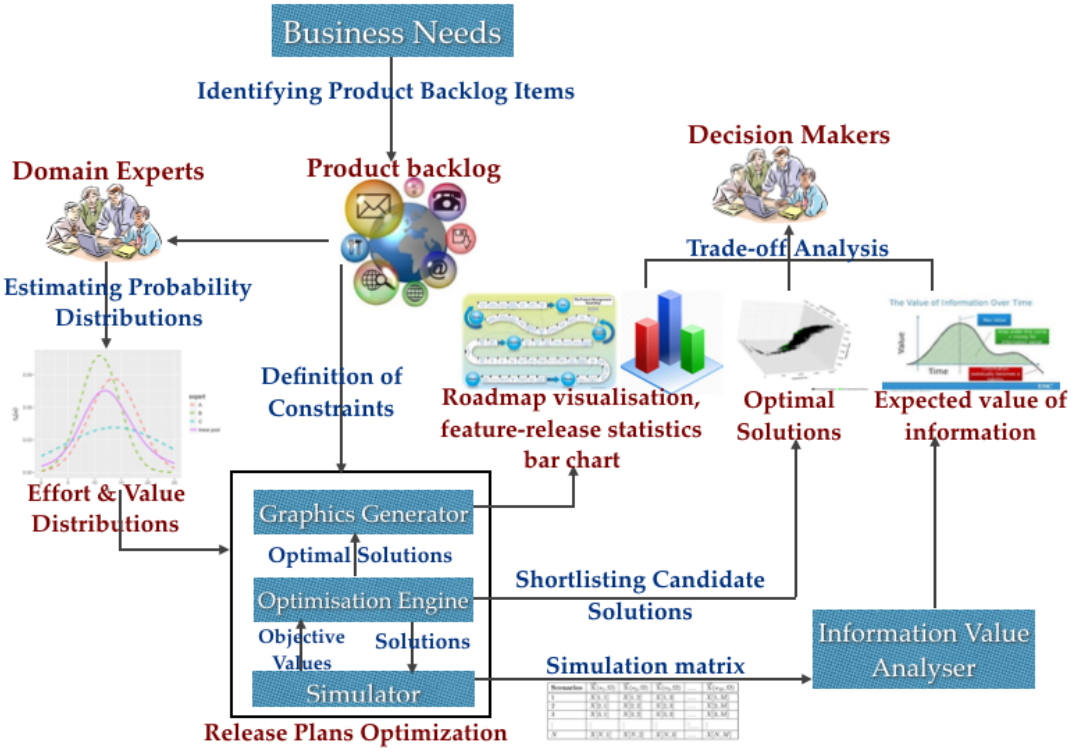


Figure 4.1: BEARS Framework

4.1 BEARS: Overview

BEARS supports release planning under uncertainty in the context of release cycles with fixed date and flexible scope. BEARS is an acronym for "Bayesian Economic Analysis of Release Scenarios". The method's name is a nod to the seminal book, "Waltzing with Bears: Managing Risks on Software Projects", one of the first books to argue for embracing uncertainty in software projects [80]. BEARS is a Bayesian method in the sense that it uses Bayesian probability to model release planners' subjective uncertainty about development effort and business value of candidate features. Like IFM, BEARS evaluates release plans by performing an economic analysis of their value flows. BEARS is different from IFM in that it estimates release plans with uncertain cash flows projections. BEARS estimates effort similar to those described in Section 2.1.3, except that BEARS work items' effort is estimated with uncertainty.

Figure 4.1 shows the sequence of activities in the BEARS framework. Each activity of the framework will be discussed in detail over the next few sections. The process of identifying product backlog items is as described in Section 2.1.1. Various techniques [81, 82, 83] have been proposed in goal modeling to derive work items from high level system goals. For example, the full product backlog of the local council project described in Section 1.1 is shown in Table 4.1. The next activity involves estimation of the effort and value of work items in the product backlog as a probability distribution. The outputs of the elicitation and estimation stages are the inputs required in the optimization stage. The optimization stage accepts product backlog items and their estimated probabilities and produce a shortlist of candidate release plans that optimizes organization objectives. BEARS presents shortlisted release plans to decision makers using different visualization tools such as scatter plot, road map view and bar chart. BEARS also computes value of information to inform decision makers about the value of gathering more information before making the final decision. To the best of our knowledge, BEARS is the first approach to compute value of information in software release planning.

The main input to BEARS is a product backlog. The product backlog in Section 2.1.1 is extended by (i) effort and value estimates are now probability distributions instead of point-based estimates, and (ii) the value of a work item is the value brought about by that work item in each release instead of its total value over some unspecified duration. The value of each item is estimated in monetary unit e.g £, $ while effort is estimated in person-hours or person-days. Given the set of work items WI and work dependencies in the product backlog where each work item w_i has the following attributes:

- *effort(w)*: a real-valued probability distribution denoting the release planners' subjective uncertainty about the number of person-days required to deliver

Work Item	Description	Depends on
1	Sign up and login	-
2	Interface to residents repository	-
3	Payment processor	-
4	Interface to rubbish collection system	-
5	Interface to Council Housing Repository	-
6	Interface to parking fine system	-
A	View council tax bills	1,2
B	Apply for council tax reduction	1,2
C	Pay council tax	A,3
D	View council tax reduction claim	B
E	Report house move	1,2
F	Apply for parking permit	1,2,3
G	Buy visitor parking permit	1,2,3
H	Pay parking and traffic fine	3,6
I	Look up rubbish collection day	4
J	Report missed rubbish collection	2,4
K	Order recycling bin	1,4
L	Submit housing application	5
M	Report accommodation problem	5
N	Submit planning application	3
O	Comment on planning application	1
P	Create application alert	N
Q	View planning applications	-
R	Contest parking fines	6
S	Set up council tax direct debit	-

Table 4.1: Full Product backlog for the Local council project

w;

- *value(w)*: a real-valued probability distribution denoting the business experts' subjective uncertainty about the value in monetary units brought about by w in each period after it is delivered.

BEARS also requires release planners to specify the following parameters:

- the planning horizon $H \in \mathbb{N}^+$
- the team capacity, noted *capacity(h)*, in each period $h \in [1..H]$;
- the investment horizon $L \in \mathbb{N}^+$

- the discount rate $r \in \mathbb{R}$ used to compute net present values;
- and, optionally, the budget B allocated to development team for the next H iterations. This is a fixed quantity to cover all costs for the fixed duration of the next H releases.

Given these inputs and parameters, BEARS evaluates candidate release plans against the following criteria:

- their Expected Net Present Value (ENPV) (described in 4.3.1), and
- their expected punctuality (EP) defined as the expected percentage of work items delivered on time (described in 4.3.2);

In the next few sections, we explain how BEARS uses these parameters for generating release plans. We explain our method for eliciting uncertainty in Section 4.2 and BEARS simulation 4.3. BEARS optimization is presented in Section 4.4 and computation of information value is explained in Section 4.5.

4.2 Estimating Effort and Value Distributions

The aim of uncertainty elicitation is to obtain real-valued distributions closely representing the knowledge and beliefs of an expert or group of experts about the effort and value of work items in the product backlog. Probability distribution is perfect for representing uncertainty about stochastic variables but eliciting probability distribution is a non-trivial elicitation process and imperfect [84]. Recent work in probabilistic elicitation have shown that experts are generally capable of providing credible judgments about a stochastic variable in their domain [72, 68].

Ideally, in order to estimate the probability distribution of a quantity of interest accurately, say Z, we need to obtain experts' probability judgments for all possible value of Z. These judgments can be infinitely large and impossible to obtain in reality. Therefore, rather than eliciting for all possible expert's probability judgment for Z, we elicit a small number of quantitative judgments from experts and then fit a convenient distribution on those judgments.

Eliciting effort and value uncertainty can be done through reliable, principled elicitation techniques for eliciting subjective uncertainty from persons and groups of persons [72]. These techniques assume that people have some real, tacit knowledge about the quantities of interest (in our case work, the items' effort and value) that can be uncovered through targeted questions and modelled as Bayesian probability distributions. The elicitation techniques are designed to mitigate common estimation biases such as overconfidence and anchoring. In BEARS we use the SHELF elicitation framework [72, 76] and the associated MATCH tool [3] to perform such elicitation. SHELF elicitation framework [72] proposed many methods to carry out this elicitation but we use the 'quartile method' in BEARS.

4.2.1 Pre-elicitation Tasks

In this phase, the elicitation facilitator, usually the product manager organizes a workshop to be attended by business and technical experts. Business experts can provide estimate of the value of work items while technical experts can provide estimates of effort needed to develop the work items. Such session will be more productive with 4-10 experts [76]. Too few number of experts might result in a biased result while too many experts might unnecessarily prolong the session. The choice of experts must be guided by the diversity of their experience in the problem domain. The session should start with the facilitator clearly and unambiguously

explaining the purpose of the session and providing necessary information about the unknown quantities (effort and value).

4.2.2 Quartile Elicitation Method

In the quartile method, the facilitator will elicit for five quantitative judgments from the experts. The facilitator elicit for the plausible lower and upper limit, lower quartile, median and upper quartile of the quantity of interest Z. Let Z represent a real-valued quantity of interest to elicit i.e. effort or value. We describe the elicitation of these five quantities below:

1. Plausible lower L and upper limit U: The first step is to ask the experts to estimate plausible lower and upper limit for Z with 95% confidence. This means that there's a small chance (5%) that the real value of Z will fall outside the provided limits. Figure 4.2 illustrates elicitation of the lower and upper limits.

$$Pr(L \leq Z \leq U) = 0.95$$

The purpose of getting these limits as a probability distribution from

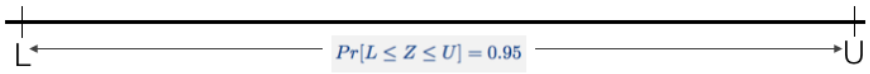


Figure 4.2: Elicitation of lower and upper bound

experts is to prevent anchoring. Anchoring is a phenomenon whereby an expert assumes too much credibility to the value they have in mind for Z and therefore provide a narrow limit for the quantity of interest. When all the experts has provided their plausible lower and upper limit for Z, The facilitator will ask questions to help the experts adjust their estimates. For example, the facilitator might randomly choose a lower limit $L0$ or upper

limit $U0$ of one of the experts and announce it to the remaining experts. If other experts are surprised about the value of $L0$, then it means $L0$ is not the true lower limit and should be adjusted but if the experts agree with $L0$, it probably means they might need to change their own limit. This process is performed until an agreement is reached about these limits.

2. Elicit median M, lower quartile $Q1$, and upper quartile $Q3$: Figures 4.3,4.4, 4.5 illustrate elicitation of the lower quartile, median, and upper quartile respectively. The facilitator elicit for the median of Z. The median M of the quantity of interest Z is a value between L and U such that it is equally likely that the true value of Z is below or above M. Median should not be misinterpreted to be the midpoint of L and U but should rather be judged on the equality of probability. M is the median of Z if:

$$Pr(Z < M) = Pr(Z > M), \ where \ L < M < U$$

The elicitation facilitator then elicits for the lower and upper quartile. The

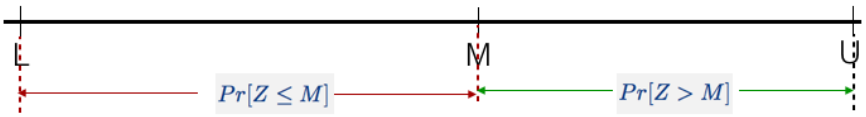


Figure 4.3: Elicitation of Median

lower quartile $Q1$ of the quantity of interest Z is a value between L and M such that it is equally likely that the true value of Z is below or above $Q1$. $Q1$ is the lower quartile of Z if:

$$Pr(Z < Q1) = Pr(Z > Q1), \ where \ L < Q1 < M$$

The upper quartile $Q3$ of the quantity of interest Z is a value between M and U such that it is equally likely that the true value of Z is below or

above $Q3$. $Q3$ is the upper quartile of Z if:

$$Pr(Z < Q3) = Pr(Z > Q3),\ where\ M < Q3 < U$$

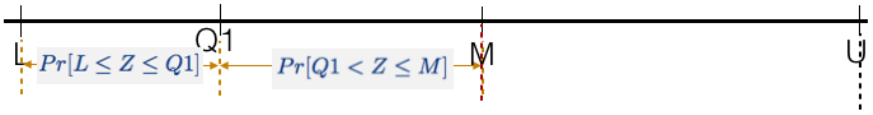


Figure 4.4: Elicitation of lower quartile.

The elicitation facilitator helps the experts to adjust their estimates for these data points by testing their indifference to alternative bets. For example, to test the stated median, the facilitator might tell the development team to imagine they will win a large sum of money if they can now correctly guess whether the real development time will be either below or above their stated median, and ask them to chose one of the two. If they have a preference for guessing either 'below' or 'above', then their stated median is not the true median of their subjective uncertainty and it should be adjusted accordingly. A similar strategy can be used to adjust the values for the lower and upper quartiles.

3. Obtaining the group judgments: It is not sufficient to just have individual judgments of experts. Due to different knowledge, expertise and interest of the experts; their opinions of the quantity of interest Z will be different and hence different distribution. We need a single distribution, but it should be one that represents a synthesis of the available knowledge and

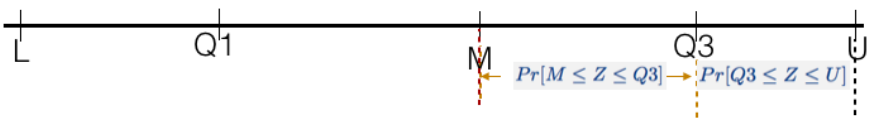


Figure 4.5: Elicitation of upper quartile.

judgment. There is no single distribution that all experts would accept as representing their true opinion. Hence, the consensus distribution will be obtained through discussion of the individual distributions among the experts while a rational impartial observer assign weights to each individual distribution based on what he or she heard during the discussions.

4. Fitting judgments to a real distribution: In this step, the elicited group judgments are fitted into a series of probability distributions matching as closely as possible to the elicited judgments. Currently, the tool supporting BEARS assumes effort and value have lognormal distributions, although we intend to support a wider range of distributions in the future. Lognormal distributions are appropriate to model uncertainty about quantities that are always positive and where the distribution can be asymmetric and have a long tail of possible high values. These characteristics fit well the typical uncertainty about software development effort and value.

As an example, Figure 4.6 illustrates the elicitation of uncertainty about the effort to develop feature A in the product backlog using the 'quartile method'. Using the same approach, the efforts and values of work items in the backlog of local council project are estimated in Table 4.3 and 4.2 respectively.

4.3 Simulating BEARS Release Plans

In order to simulate release plans, BEARS distinguishes *release plans* $p : WI \rightarrow [1..K]$ that specify when work items are planned for release from *release scenarios* $s : WI \rightarrow [1..K]$ that specify when work items are actually released. During release planning, the future actual release scenario is unknown. To simulate release scenarios, BEARS first generates N simulations of work items' effort and value drawn from the effort and value probability distributions. By default

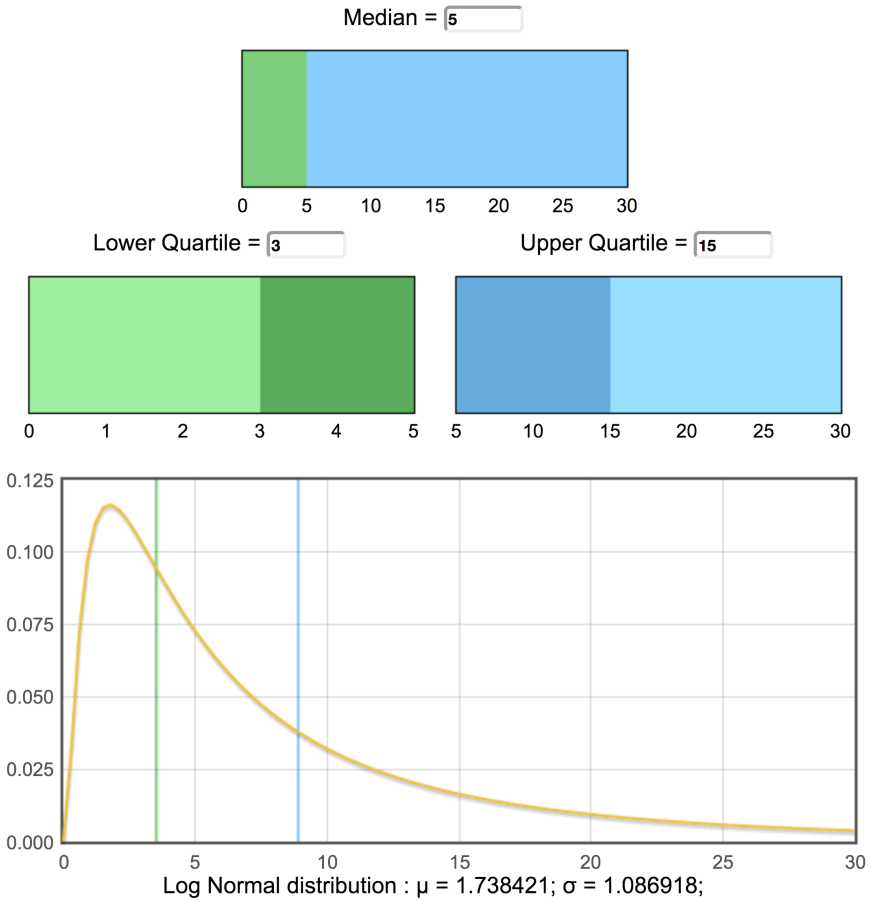


Figure 4.6: Eliciting effort uncertainty for feature "A: view council tax bills" with the MATCH tool [3].

BEARS uses $N = 10^4$. This creates N possible future worlds, each having specific effort and value data for all work items. In each future worlds, BEARS creates the release scenario that will happen in this world based on the work item's effort data. This is done by first mapping the release plan p to a work sequence ws that specifies in what order work items in the plan will be worked on (note that release plan does not specify any order between items to be delivered in the same release), then generate the release scenarios from the work sequence.

To generate a work sequence ws from a release plan, we apply the following priorities between work items. Let $meanValue(w_i)$ be the mean value and $meanEffort(w_i)$ be the mean effort of work item w_i over N simulated scenarios of effort and value, then the ratio of mean value to mean effort of work item

Work Item	Log-Normal Distribution $lnN(\mu, \sigma^2)$
A	$\mu = 2.51,\ \sigma = 1.04$
B	$\mu = 2.97,\ \sigma = 0.75$
C	$\mu = 2.51,\ \sigma = 1.04$
D	$\mu = 2.97,\ \sigma = 0.75$
E	$\mu = 2.51,\ \sigma = 1.04$
F	$\mu = 3.42,\ \sigma = 0.65$
G	$\mu = 3.55,\ \sigma = 0.67$
H	$\mu = 3.02,\ \sigma = 1.05$
I	$\mu = 2.22,\ \sigma = 0.82$
J	$\mu = 2.09,\ \sigma = 0.82$
K	$\mu = 2.51,\ \sigma = 1.04$
L	$\mu = 3.13,\ \sigma = 1.03$
M	$\mu = 2.73,\ \sigma = 0.65$
N	$\mu = 3.09,\ \sigma = 1.02$
O	$\mu = 3.33,\ \sigma = 0.71$
P	$\mu = 1.66,\ \sigma = 0.90$
Q	$\mu = 3.42,\ \sigma = 0.65$
R	$\mu = 1.66,\ \sigma = 0.90$
S	$\mu = 2.05,\ \sigma = 1.59$

Table 4.2: Elicitation of Value Distributions for the Local Council running example.

w_i denoted $VERatio(w_i)$ is defined as,

$$VERatio(w_i) = \frac{meanValue(w_i)}{meanEffort(w_i)}$$

A work item w_i takes precedence over another work item w_j in the work sequence if:

i w_i is planned for an earlier release than w_j (i.e. $p(w_i) < p(w_j)$), or

ii w_j has a precedence dependency on w_i (i.e. $w_i \leftarrow w_j$), or

iii both items are planned for the same release and have no precedence dependency and $VERatio(w_i) > VERatio(w_j)$

From the work sequence, we assign work items to release periods such that the d^{th} item in the work sequence is delivered in release k if, and only if, the cumulative

Work Item	Log-Normal Distribution $lnN(\mu, \sigma^2)$
1	$\mu = 2.05,\ \sigma = 0.59$
2	$\mu = 2.61,\ \sigma = 0.42$
3	$\mu = 2.45,\ \sigma = 0.50$
4	$\mu = 2.61,\ \sigma = 0.42$
5	$\mu = 2.61,\ \sigma = 0.42$
6	$\mu = 2.61,\ \sigma = 0.42$
A	$\mu = 1.75,\ \sigma = 0.73$
B	$\mu = 2.61,\ \sigma = 0.65$
C	$\mu = 1.88,\ \sigma = 0.64$
D	$\mu = 1.75,\ \sigma = 0.73$
E	$\mu = 2.35,\ \sigma = 0.65$
F	$\mu = 2.61,\ \sigma = 0.65$
G	$\mu = 2.21,\ \sigma = 0.75$
H	$\mu = 2.32,\ \sigma = 0.88$
I	$\mu = 1.98,\ \sigma = 0.71$
J	$\mu = 1.98,\ \sigma = 0.71$
K	$\mu = 1.98,\ \sigma = 0.71$
L	$\mu = 2.61,\ \sigma = 0.65$
M	$\mu = 1.98,\ \sigma = 0.71$
N	$\mu = 2.98,\ \sigma = 0.38$
O	$\mu = 2.05,\ \sigma = 0.59$
P	$\mu = 1.37,\ \sigma = 0.38$
Q	$\mu = 1.37,\ \sigma = 0.38$
R	$\mu = 2.21,\ \sigma = 0.75$
S	$\mu = 1.30,\ \sigma = 0.69$

Table 4.3: Elicitation of Effort Distribution.

effort to develop the work sequence up to item d is more than the cumulative team capacity up to release period $k-1$ and less or equal to the cumulative team capacity up to release period k:

$$\sum_{j=1}^{k-1} capacity(j) < \sum_{j=1}^{d} effort(ws(j)) \leq \sum_{j=1}^{k} capacity(j)$$

where $ws(j)$ is the j^{th} element in the work sequence ws. This condition ensures that the total effort in each release period does not exceed the team capacity for that period and that work items are released in order of the work sequence derived from the release plan. Figure 4.7 shows the algorithm for generating a

release scenario from work sequence.

Given work sequence, ws
1. Let $cumulativeEffort = 0$
2. Let $currentIteration = 1$
3. $cumulativeCapacity = capacity(currentIteration)$
4. for $i = 0$ *to* $length(ws)$
   5. $cumulativeEffort+ = effort(ws(i))$
   6. while($cumulativeEffort$ is greater than $cumulativeCapacity$)
      7. Increase $currentIteration$ by 1
      8. Increment $cumulativeCapacity$ by $capacity(currentIteration)$
      i.e. $cumulativeCapacity \mathrel{+}= capacity(currentIteration)$

   9. $actualReleasePeriod(ws(i)) = currentIteration$

Figure 4.7: Algorithm for generating release scenario from a work sequence.

4.3.1 Evaluating Expected Net Present Value

BEARS computes the expected net present value of a release plan p as the mean net present value of the N release scenarios that simulate p. In a release scenario s, let WI_k be the work items assigned to release k, then the total value delivered during release period k is the sum of the value delivered by all the work items:

$$value(WI_k) = \sum_{w_i \in WI_k} value(w_i, s, k)$$

, where $value(w_i, s, k)$ is the value delivered by a work item w_i in period k for a given scenario s and defined as:

$$value(w_i, s, k) = \sum_{l=k}^{L} \frac{value(w_i, s, l)}{(1 + \frac{r}{100})^l}$$

The Net Present Value (NPV) of release scenario s is then defined as:

$$NPV(s) = \sum_{i=1}^{K} value(WI_k) - B \tag{4.1}$$

where B is the budget allocated to the team for the periods in the planning horizon. If no budget is specified, BEARS uses $B = 0$.

The net present value of a release plan is a random variable whose distribution is approximated by the net present value of all release scenarios simulated from the release plan.

$$ExpectedNPV(p) = \sum_{s \in N} \frac{NPV(s)}{N} \tag{4.2}$$

4.3.2 Evaluating Expected Punctuality

The punctuality of a release scenario s with respect to a release plan p is defined as the percentage of planned work items delivered on or before their planned release period:

$$Punctuality(s, p) = \frac{\#\{w \in WI \mid s(w_i) \leq p(w_i)\}}{\#dom(p)}$$

where *dom(p)* is the domain of p, i.e. the set of work items in the release plan.

The expected punctuality of a release plan p is the mean punctuality of the N release scenarios that simulate p:

$$ExpectedPunctuality(p) = E(Punctuality(s,p)) \tag{4.3}$$

Our punctuality metric is motivated by the need for a simple way to communicate uncertainty about the scope of future releases in a release plan. A release plan with a 90% expected punctuality means that 90% of work items are expected to be

delivered in their planned release. This expected punctuality metric is motivated by the need for a simple metric to help release planners and software development teams communicate the uncertainty associated with different release plans to clients and other stakeholders. The metric also allows release planners to compare release plans with different expected punctuality and analyse tradeoffs between expected punctuality and expected NPV, as explained in the next section.

4.4 Shortlisting Release Plans

BEARS requires release planners to specify additional parameters such as planning horizon K, investment horizon L, discount rate r, efforts constraints and optionally budget constraints for the next K planning horizons. For example, assume discount rate $r = 2\%$ per release period, number of releases $K = 3$, investment horizon $L = 12$, $capacity = 40$ person-days per release.

BEARS shortlists a set of Pareto-optimal solutions that maximize expected net present value ($ENPV$) and expected punctuality (EP). In BEARS, a release plan p is Pareto-optimal if there is no other release plan p' that outperforms p on one criteria and is at least as good for the other criteria i.e. such that either $ENPV(p) > ENPV(p')$ and $EP(p) \geq EP(p')$ or $EP(p) > EP(p')$ and $ENPV(p) \geq ENPV(p')$.

In general, the number of release plans will be too large to compute the exact set of Pareto-optimal solutions. BEARS thus relies on multi-objective evolutionary algorithms (MOEAs) to explore the space of candidate release plans and generate a good approximation of the set of Pareto-optimal release plans.

Our implementation uses the following MOEAs: NSGA-II [49], MOCell [85], and SPEA2 [50]. To apply such algorithms, BEARS encodes a release plan p as an

array of integers where each element represents the release number $p(w)$ assigned to a work item or zero if the work item is not scheduled in the plan. The size of the array is equal to the number of the work items in the product backlog. The MOEAs start by randomly generating an initial population of 100 release plans, then iteratively evolve the population towards release plans with higher $ENPV$ and EP using genetic selection, crossover and mutation. Our implementation uses integer simulated binary crossover with probability $P_c = 0.9$ and polynomial mutation with probability $P_m = \frac{1}{|WI|}$ where $|WI|$ is the number of work items. We have set the MOEAs to terminate after having explored $25,000$ release plans. On termination, they return the Pareto-optimal release plans in their final population.

During mutation and crossover, the MOEAs may generate new release plans that violate the dependency constraints between work items. Following an approach used in previous work, our implementation automatically detects and repairs such violations so that the populations only contain valid release plans [86]. If in the future, BEARS is extended to support more dependency relations between work items, such as coupling, exclusion and weak precedence, the only required change will be to extend the detection and repair of constraints violations during mutation and crossover.

When the search terminates, BEARS displays the Pareto-front of the shortlisted release plans. Visualizing such Pareto-front helps release planners analyze and communicate trade-offs between expected value and punctuality.

BEARS generates different views of the solutions in order to support the decision makers in making optimal release plan decisions. We generate visualizations of the candidate release plans using table, scatter plot and bar chart to help decision makers in deciding what plan to select. The scatter plot provides information about the regions of the solution space containing the best solutions and how much of the solution space was explored by the optimization algorithm. The table

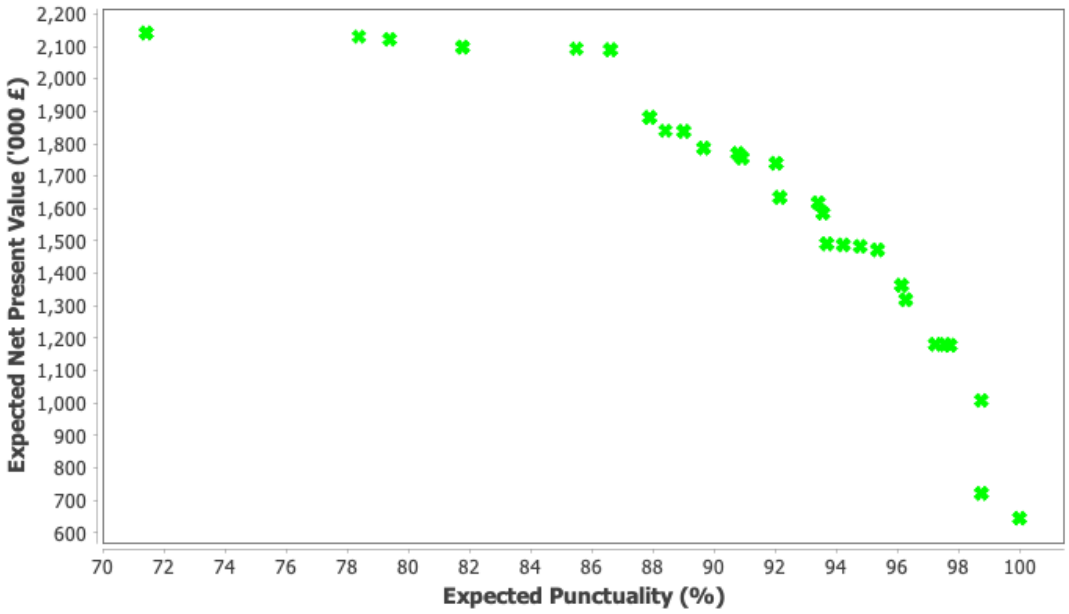


Figure 4.8: Expected NPV and punctuality of shortlisted release plans for flexible-scope release

view shows details of work item assignment to each release and corresponding objective value for each plan. The bar chart view provides a summary of the shortlisted plans by showing the frequency of each work item in each of the release. The cash flow analysis view of the release plans gives possible financial projection of each release plans if chosen. The analysis answers questions such as when will the project the break-even if a certain release plan is chosen? When will self-funding status be achieved? What will be the company financial position in period t if a particular release plan is chosen? Finally, BEARS calculates the expected value of perfect information on the generated shortlist so as to help release planners determine whether or not to seek more information to reduce their uncertainty before making release decisions. The computation of information value are described in Section 4.5.

For example, Figure 4.8 shows the returned Pareto-front for our local government problem. Visualising such Pareto-front helps release planners analyse trade-off between expected value and punctuality. The figure shows that the release plan

with the highest *ENPV*, on the top left, has an expected punctuality around 70%. Release planners may hesitate selecting a release plan where 30% of work items are likely to be delivered late. Late deliveries, even if forewarned, will disappoint stakeholders and could put the project at risk of being cancelled. Developers' morale and productivity could also be affected. To address these issues, release planners could select a release plan with slightly lower *ENPV* but higher expected punctuality up to 86%. Release plans with still higher expected punctuality are possible but would require further sacrifice in *ENPV*. If the client requires a punctuality of 95% or even insists on 100% punctuality (e.g. with penalties if the targets are not met), release planners will prefer selecting release plans that are easier to deliver on time. Selecting a plan with an expected punctuality of 95% requires reducing *ENPV* to around £1.5M and a plan with 100% expected punctuality requires reducing *ENPV* to around £600K. BEARS output therefore allow release planners to have informed discussions with clients and developers about tradeoffs between *ENPV* and expected punctuality.

4.5 Information Value Analysis

Uncertainty arises due to lack of complete information about parameter of interests before making decision. The presence of uncertainty in the process of making business decisions making business decisions means that there's likelihood of making incorrect decisions [68]. By incorrect decision, we mean that decision makers may have selected another alternative if some information was available at the time decision was made. Therefore, additional information can (i) reduce uncertainty about decisions that have economic consequences (ii) affect the behavior of decision makers in choosing an alternative course of action. If it is possible for decision makers to pay for additional information that reduces uncertainty before making decisions, how much will that information be worth to

them? However, additional information are worthwhile if and only if the cost of the new information is lower than the value derived from the new information [41, 68, 73]. The question can be answered through information value analysis. Information value analysis computes the upper bounds of the financial value of additional information in decision problems.

The expected value of information is defined as the expected gain in benefit between selected alternatives with or without additional information [68, 87]. Expected value of information for the different uncertain variables (i.e. value of a work item) guides decision makers to avoid wasting resources on reducing uncertainty about work items of low value but rather focus on reducing uncertainty about work items with high expected value [41]. The expected value of information is defined with respect to an outcome to be maximised and assumes a default decision strategy of maximising expected benefit. In BEARS, the outcome to be maximised is the net present value (NPV) but the definition applies to any other benefits. Expected value of total perfect information (EVTPI) is the expected gain in net present value from using perfect information about value of all work items.

$$EVTPI = E[\max_{a \in A} NPV(a, \phi)] - \max_{a \in A} E[NPV(a, \phi)],$$

where A is the set of alternative release plans, a is a single alternative in A, $NPV(a, \phi)$ is the net present value of alternative a given the uncertain work items values ϕ.

Similarly, the expected value of partial perfect information about the value parameter of a single work item θ denoted $EVPPI(\theta)$, is the expected gain in net present value from using perfect information about θ [88].

$$EVPPI(\theta) = E[\max_{a \in A} f(a, \delta)] - \max_{a \in A} E[NPV(a, \phi)],$$

where Φ is a set of uncertain work items parameter, $f(a, \delta) = E_{\Phi - \theta} NPV(a, \phi)$ is the expected NPV of alternative a, give that the parameter θ is fixed at δ and $E_{\Phi - \theta}$ is the expectation with respect to all model parameters Φ except θ.

The information value analyser in BEARS accepts the Pareto optimal solutions and their corresponding NPV values as input and returns $EVTPI$ and $EVPPI$. Due to the large size of the solution space, it is not feasible to compute information value for all solution alternatives. Therefore, we define A as the set of Pareto release plans returned from BEARS optimization phase. This assumption is reasonable because it will be a waste of time to perform further analysis on non-shortlisted release plans.

4.6 BEARS **JAVA Tool**

BEARS is supported by a tool that automatically shortlists a set of Pareto-optimal release plans that maximize $ENPV$ and expected punctuality through Monte-Carlo simulation and search-based multi-objective optimization. The tool is implemented in JAVA. It uses JMetal [89] for multi-objective optimization and our own implementation of a Monte-Carlo simulation of release plans. The tool takes as input a CSV file defining the work items' dependencies and effort and value estimates. It generates a CSV file with the shortlisted Pareto-optimal release plans and a figure of the Pareto front graph. It also computes and reports the expected value of perfect information, a form of probabilistic sensitivity analysis used to help release planners decide whether they should seek more information about some item's effort and value before making a decisions [41].

4.7 BEARS Limitations

BEARS was developed to support our experiments. For practical use, the method has a number of limitations.

Applicability and cost. As mentioned earlier, we have not yet evaluated the method in an industrial context. As a result, the applicability and cost of BEARS in industrial context are still unknown. Other release planning methods under uncertainty suffer from the same limitation. In contrast, EVOLVE-II and some other methods that ignore uncertainty have been applied in industry and their benefits documented [11, 12, 13]. We hope the results of our experiments will be used to justify future industry trials of release planning methods under uncertainty.

Modelling assumptions. The economic model used in BEARS (Eq. 4.1 to 4.3) relies on simplifying assumptions that may affect the accuracy of their net present value and punctuality estimations. Notably, the model assumes that work items are independent. In practice, the values of two work items may depend on a third variable making this assumption invalid (e.g. the values of online services related to council tax payments all depend on the number of residents who pay council taxes). This limitation can be addressed by developing more complex models but this would make the release planning method more difficult to apply as the models would need to be project-specific.

Tool limitations. The BEARS tool is a prototype developed primarily to support our experiments. The tool supports the evaluation and shortlisting of release plans (Sections 4.3 to 4.4) but does not support uncertainty elicitation (Section 4.2) and does not include features to help release planners explore the shortlist and select their preferred plan. Such features are essential for future industrial applications.

4.8 Chapter Summary

BEARS is the first release planning method that supports the industrial practice of fixed-date, flexible-scope releases.

As future work, we plan on extending BEARS to analyse a wider range of criteria and work items during release planning such as: analysing fairness and multiple value dimensions from multiple perspectives [30]; managing technical debt during release planing [90]; supporting decisions about what experiments to perform (e.g. as A/B tests) and data to collect as part of the release planning process and to support future release planning decisions; and analysing objective data (e.g. from the software development process, software usage, and users' feedback) to update uncertainty about feature's effort and value (using Bayes' rule) so as to inform future decisions [91].

Chapter 5

Evaluation

We conducted three experiments to evaluate and compare BEARS to existing release planning methods. The first compare BEARS to EVOLVE-II on our local government release planning problem. The second extends the first by comparing BEARS to more release planning methods on a larger number of release planning problems. The third evaluates the performance of the alternative MOEAs used in BEARS (NSGA-II, SPEA2, and MOCell).

The first two experiments aim to address fundamental research questions we raised in the introduction chapter (Section 1.3):

- Does analysing uncertainty during release planning lead to better release planning decisions than if uncertainty is ignored? Would different release planning methods applied in the same context recommend the same release plans? If not, do some methods make better recommendations than others?

In these experiments, we will apply BEARS and other release planning methods on a set of release planning problems and compare the resulting shortlisted release plans. We will observe that release plans shortlisted by BEARS are better than

those shortlisted by other methods in the sense that they have higher expected net present value and expected punctuality.

In the previous chapters, we have argued that expected net present value and expected punctuality are better criteria for shortlisting release plans than the simpler net present value (or value points) used by methods that ignore uncertainty. Evaluating expected net present value is more accurate than evaluating net present value (or value points) because it takes into account that work items may be delivered later than planned. Evaluating expected punctuality is more informative and realistic than assuming all work items will be delivered on time. We have therefore argued that optimising expected net present value and expected punctuality is preferable to optimising net present value (or value points) without considering punctuality. So far, we have argued that analysing uncertainty is in theory better than ignoring it. In this chapter, we study whether analysing uncertainty also makes a difference in practice.

5.1 Experiment I: BEARS vs. EVOLVE-II

- RQ1: How does BEARS shortlists compare to EVOLVE-II when applied on the same backlog?

Our first experiment compares BEARS to EVOLVE-II on our motivating example. We have chosen EVOLVE-II because it epitomizes the use of value points in release planning. It is also the most prominent release planning method in literature [9, 10], is supported by a professional tool, and has been applied in industrial contexts [11, 12].

The first objective of this experiment is to provide a concrete illustrative example of the differences between release plans shortlisted by BEARS and EVOLVE-II.

The second objective is to introduce our approach to comparing alternative release planning methods on the same problem. The same approach will be used in our second experiment, and introducing this approach on a single example will make the second experiment easier to explain.

Our first and second experiments aim to compare alternative release planning methods when the models are used on the same product backlog. Alternative models use different optimisation criteria. For example, BEARS has two objectives, maximizing *ENPV* and maximising expected punctuality, whereas EVOLVE-II has a single objective, maximising value points. Alternative models also require eliciting different inputs from stakeholders. For example, BEARS requires eliciting probability distributions for the work items effort in person-days and value in £, whereas EVOLVE-II requires eliciting work items effort in person-days and value points. However, when applied to the same product backlog, different models evaluate and shortlist release plans from the same set of release plans (they have the same solution space). The purpose of our experiments are to compare shortlists generated from different models. Does BEARS shortlist the same release plans as EVOLVE-II and other methods? If not, how significant are the differences between the shortlists?

5.1.1 Experiment Design

Chapter 4 presented the application of BEARS on our running example. The shortlisted release plans were shown in Figure 4.8.

Imagine now that we had used EVOLVE-II instead of BEARS in the same situation. Starting from the same product backlog described in Section 1.1, we would have estimated the work items' effort and value as described in Section 2.1.3 and 2.1.4. This would have resulted in estimates such as those shown Table 2.3.

We would also have had to define the release weights and capacities. We would then have used ReleasePlanner, the tool supporting EVOLVE-II, to shortlist *top 5* release plans that maximize value points (Eq. 2.4) subject to capacity constraints (Eq. 2.5).

Our experiment will simulate this process of applying EVOLVE-II. Instead of eliciting effort and value estimates from stakeholders, we will derive such estimates from the BEARS estimates by assuming that the stakeholders' EVOLVE-II estimates would be consistent with the estimates they have given using BEARS. This allows us to compare the outputs of BEARS and EVOLVE-II optimisation models independently from differences between their effort and value elicitation techniques. For our experiments, we set the EVOLVE-II effort and value point estimates to be the mean of the value and effort probability distributions in BEARS, normalised on a scale from *0* to *9*. We chose a scale from *0* to *9* because it is the default scale in EVOLVE-II. Likewise, we define other inputs of the EVOLVE-II model so that they are consistent with the corresponding parameters in the BEARS model. We define the team capacity for each release period in EVOLVE-II from the team capacity in the BEARS model using the same approach used to scale effort estimate. We set the weight of each release period i to be $(1+r)^{-i}$ where r is the NPV discount rate in the BEARS model. This ensures that both models apply the same relative weights to sum up values in different release periods.

Once all model inputs have been defined, we have used ReleasePlanner 2.0, the commercial tool supporting EVOLVE-II, to shortlist release plans with high value points among those that satisfy the capacity constraints. ReleasePlanner uses a specialised integer programming algorithm that shortlists release plans that can be proven to be within a given bound of the maximal value points [6, 92]. The algorithm also uses heuristics to increase diversity (in the solution space) among

the shortlisted release plans. This approach recognises that the EVOLVE-II model, like any model, is necessarily approximate, and it gives release planners a more diverse set of choices than simply selecting the release plan with the most value points.

In order to compare optimal release plans in the BEARS and EVOLVE-II models without being affected by the optimisation algorithm, we have developed an alternative optimisation algorithm for EVOLVE-II that uses the same MOEAs with repair as BEARS. In this example, we have used NSGA-II. The objective function here is to maximize value points. The algorithm shortlists the top n release plans that satisfy the capacity constraints, without considering diversity among the shortlisted plans. We call this approach EVOLVE-*max*. When comparing BEARS to EVOLVE-*max*, we set the EVOLVE-max shortlist to be of the same size of the BEARS shortlist (28 in this example). ReleasePlanner, in contrast, shortlists 5 release plans only and we did not have control over this number. On this example, EVOLVE-*max* shortlists release plans with higher value points than ReleasePlanner: from *331* to *361* value points for EVOLVE-*max*, from *335* to *341* for ReleasePlanner.

After generating the ReleasePlanner and EVOLVE-max shortlists, we compute the expected NPV and expected punctuality of all shortlisted release plans using the BEARS model (Equations 4.1-4.3) so that they can be compared to the release plans shortlisted by BEARS.

5.1.2 Results

Figure 5.1 shows the expected NPV and expected punctuality of release plans in the BEARS, ReleasePlanner and EVOLVE-*max* shortlists. For the release plans in the EVOLVE-*max* shortlist, we have used the BEARS model to compute their

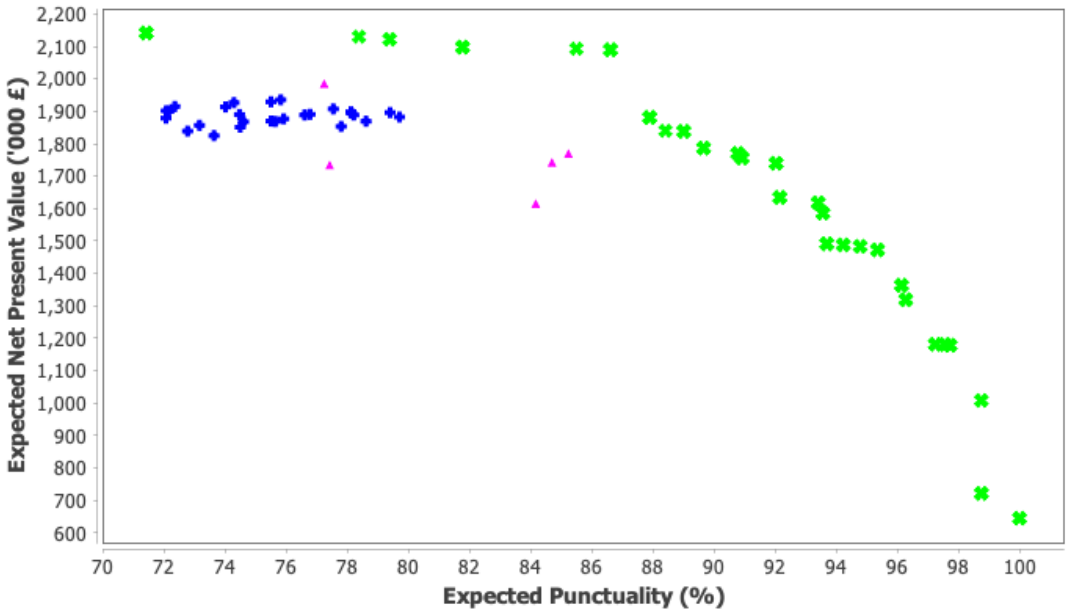


Figure 5.1: Release plans shortlisted by BEARS(green crosses), ReleasePlanner (red triangle) and EVOLVE-*max* (blue diamonds) for the local government project.

expected net present value and expected punctuality. For a fair comparison, we include as many release plans in the EVOLVE-max shortlist as there are in the BEARS shortlist (*27* in this example). Figure 5.1 shows that:

1. The shortlists are disjoint; none of the release plans shortlisted using ReleasePlanner or EVOLVE-*max* are short-listed by BEARS, and vice-versa.

2. The BEARS shortlist *strictly dominates* the two EVOLVE-II shortlists; in other words, for every release plan in the EVOLVE-II shortlists there is at least one release plan in the BEARS shortlist that has higher expected *NPV* and higher expected punctuality.

3. The maximum expected *NPV* is nearly 10% higher in the BEARS shortlist than in the EVOLVE-II shortlists. The difference is due to the EVOLVE-II model evaluating value points under the assumption that all work items are delivered on time. Hence, the release plans with the most value points are not necessarily those with the highest expected *NPV*.

4. The expected punctuality in the BEARS shortlist ranges from 72% to 100%. In contrast, the highest expected punctuality in the EVOLVE-II shortlist is around 80%. Most importantly, in practice ReleasePlanner users would not be aware of the release plans' expected punctuality.

In summary, Figure 5.1 shows that on this example BEARS shortlists release plans that are better than those shortlisted by EVOLVE-II in terms of expected *NPV* and expected punctuality. BEARS also provide information about expected punctuality that is missing in ReleasePlanner.

Incidentally, Figure 5.1 also shows that the EVOLVE-II strategy of shortlisting diverse release plans is useful on this example because it shortlists release plans with a better range of expected punctuality than EVOLVE-*max*. This usefulness is however reduced by the lack of information that ReleasePlanner users would not have about expected punctuality.

In terms of run-time, BEARS is slower than EVOLVE-II. In this example, BEARS shortlists release plans in 35 seconds, compared to around 3 seconds for EVOLVE-*max* and 7 seconds for the web-based ReleasePlanner (whose run-time includes GUI functions and network communications not included in EVOLVE-*max*). In our implementations, BEARS and EVOLVE-*max* are configured to stop after evaluating a fixed number of release plans (25,000). The run-time increase in BEARS is due to the Monte-Carlo simulation required to evaluate each candidate release plans.

5.1.3 Threats to Validity

External validity. This experiment compares BEARS to EVOLVE-II on a single example. The next section will extend this comparision to more release planning problems.

Internal validity. Comparing BEARS against ReleasePlanner and EVOLVE-*max* allows us to check that differences between the BEARS and EVOLVE-II shortlists are not due to differences in optimisation algorithms. We have evaluated the release plans shortlisted by EVOVLE-II in terms of the BEARS objectives of maximizing expected NPV and expected punctuality, even though such objectives are not used by EVOLVE-II. Doing so is appropriate because our objective is to study differences between release plans shortlisted by BEARS and EVOLVE-II for the local government release planning problem where two important stakeholders' concerns are the financial implications and punctuality of release plans. The BEARS model provides one way to model these two concerns. The EVOLVE-II model provides another, lighter way to model the financial concern using value point and ignores the punctuality concern due to the method's lack of support for probabilistic reasoning. This experiment showed that using the simpler EVOLVE-II model comes at a cost of shortlisting release plans with lower expected NPV and punctuality.

In this experiment, we have simulated the application of EVOLVE-II by deriving its inputs from the BEARS model instead of applying EVOLVE-II independently from BEARS. This allowed us to minimize the difference between the application of the two methods. If EVOLVE-II had been applied independently from BEARS, it would most likely have led to different effort and value estimates and probably even larger differences between the shortlisted release plans.

We have used EVOLVE-II's default 9 point scale for estimating value. Using a more precise scale might lead to different results. Our next experiment will address this concern by comparing BEARS to a method we call BEARS-*deterministic* where the mean value estimates in BEARS are not scaled down to a 9 point scale.

Finally, this experiment compared a single run of BEARS to a single run of EVOLVE-II. Since both methods use a stochastic optimisation algorithm, different

runs may generate different shortlists. Our second experiment will address this concern by repeating each experiment 30 times.

5.2 Experiment II: Comparing BEARS to Other Release Planning Methods

In our second experiment, we want to study more systematically whether analysing uncertainty using BEARS leads to shortlisting different release plans than if uncertainty is ignored. We also want to study differences between BEARS and release planning models that assume fixed-scope releases. Our research questions are:

- RQ1: How does BEARS compare to release planning methods that ignore uncertainty?
- RQ2: How does BEARS compare to release planning methods under uncertainty with fixed-scope releases?

To study these questions, we compare BEARS to 4 release planning methods on 32 release planning problems drawn from 8 product backlogs.

5.2.1 Experiment Design

Release Planning Methods

Table 5.1 lists the release planning methods in our experiment. The first two, EVOLVE-*max* and BEARS-*deterministic*, are used to answer Q1.

Table 5.1: Release Planning Methods in Our Empirical Evaluation

Method	Effort estimates	Value estimates	Simulation Method	Optimisation problem
EVOLVE II	person-days	value points	deterministic	Maximise value points subject to capacity constraint
BEARS-*deterministic*	person-days	economic value	deterministic	Maximise NPV subject to capacity constraint
EVOLVE-*with-uncertainty*	uncertain story points	value points	stochastic, fixed-scope	Maximise value points; Maximise on-time probability
BEARS-*fixed-scope*	uncertain person-days	uncertain economic value	stochastic, fixed-scope	Maximise expected NPV; Maximise on-time probability
BEARS	uncertain person-days	uncertain economic value	stochastic, flexible-scope	Maximise expected NPV; Maximise expected punctuality

EVOLVE-*max* was introduced in our first experiment. It is a variant of EVOLVE-II that uses the same release planning model as EVOLVE-II and a simpler optimization algorithm. Comparing BEARS to EVOLVE-*max* allows us to study differences between the BEARS and EVOLVE-II models, independently from the effects of the diversity heuristics used in the ReleasePlanner implementation. Our objective is to study differences between the methods' release planning models, not between their optimisation algorithms.

BEARS-*deterministic* is a variant of BEARS where effort and value are represented as point estimates instead of probability distributions. In BEARS-*deterministic*, effort are represented in person-days while values are represented using financial metrics. The optimisation problem of BEARS-*deterministic* is to maximise NPV (Eq. 4.1) subject to effort capacity constraints (Eq. 2.5). Comparing BEARS to BEARS-*deterministic* allows us to isolate the effect of modelling uncertainty from other differences between BEARS and EVOLVE-*max* (in particular the 9-point scale used in EVOLVE to estimate story points and value points). For both EVOLVE-*max* and BEARS-deterministic, our experiment will shortlist the top n release plans, where n is the maximum number of release plans in the BEARS shortlist.

The other two methods in Table 5.1, EVOLVE-*with-uncertainty* and BEARS-*fixed-scope* are used to answer Q2. EVOLVE-*with-uncertainty* is based on a previously published variant of EVOLVE that models uncertainty about effort but not value [15]. The method simulates effort uncertainty assuming fixed-scope releases and the optimisation problem has two objectives: maximise value points

Table 5.2: Release Planning Problems in Our Empirical Evaluation

Release Planning Problem	Backlog Size	Number of Constraints	Original effort estimates	Original value estimates
Local Government Project	20	23	uncertain person-days	uncertain economic value
Release Planner	25	11	person-hours	value points
Word Processor	50	65	person-hours	value points
RALIC	143	0	person-hours	value points
Synthetic-30	30	11	uncertain person-days	uncertain economic value
Synthetic-50	50	15	uncertain person-days	uncertain economic value
Synthetic-100	100	30	uncertain person-days	uncertain economic value
Synthetic-200	200	51	uncertain person-days	uncertain economic value

(Eq. 2.4) and maximise the probability of delivering all features on time (Eq. 2.6). BEARS-*fixed-scope* is a variant of BEARS that uses the same uncertain effort and value estimates as BEARS but simulates effort uncertainty assuming fixed-scope releases instead of flexible-scope. Comparing BEARS to BEARS-*fixed-scope* allows us to study the difference between fixed-scope and flexible-scope simulations independently from other factors. The shortlists in EVOLVE-*with-uncertainty* and BEARS-*fixed-scope* are the set of Pareto-optimal solutions returned by the multi-objective optimisation algorithm. Shortlists of different methods can therefore be of different sizes.

In our experiments, we have used our own implementation of each method. As stated in Section 5.1, EVOLVE-*max* uses the sames MOEAs with repair as BEARS. The variants of EVOLVE that deal with uncertainty have no publicly available implementation; we therefore had to build our own. For consistency, we have tested all methods using NSGA-II as MOEA. This reduces the risk that differences between the methods' shortlists are due to the MOEA rather than due to more fundamental differences in the optimisation models.

Release Planning Problems

Table 5.2 lists the 8 product backlogs in our experiment. For each product backlog, we consider 4 release planning problems by varying the planning horizon from 2 to 5 periods. The local government project is the illustrative example

we have used throughout the thesis, Release Planner, Word Processor, and RALIC product backlogs have been used in previous studies to evaluate the performance of MOEAs on release planning problems [24, 27, 93, 26, 94]. The product backlogs for the Release Planner and Word Processor projects contains candidate features for future releases of the Release Planner tool and for a word processor, respectively. The features were elicited and evaluated during exploratory studies for a variant of the EVOLVE method [55]. The product backlog for the RALIC project includes requirements for a future building access control system at University College London. The requirements were elicited from and evaluated by 87 stakeholders using an online tool that leveraged stakeholders' relationships to drive the requirements elicitation and prioritization process [95]. The last 4 product backlogs are synthetic backlogs of size 30, 50, 100, and 200, respectively. The work items precedence constraints, effort and value estimations have been generated at random.

For the Release Planner, Word Processor and RALIC product backlogs, the original effort and value estimates do not include uncertainty. To be able to apply BEARS on these problems, we have artificially added uncertainty to the estimates by simulating the uncertainty elicitation process described in Section 4.2. For an original point-based effort estimate x, we have set the upper and lower bounds to x and $1.8x$ and the lower quartile, median and upper quartile to $1.2x$, $1.5x$ and $1.7x$. This simulates a situation where the original effort estimate is the most optimistic value and the true development time could take up to 1.8 times this optimistic value. This situation is consistent with studies of software estimations that show that people tend to underestimate development time [38]. For an original value point estimate y, we have set the lower and upper bounds to £0 and £$1000y$ and the lower quartile, median and upper quartile to £$200y$, £$500y$, and £$750y$. This simulates a situation where one value point is worth £$1,000$ and the original value estimate is the most optimistic value and the true value could be as low as zero.

This is consistent with observations of software project where initial prediction of business value tend to be overestimated [96, 77] We make no claim that the values we have chosen represent the true uncertainty that would have been elicited from real developers and stakeholders in these projects. Since our objective is to compare the outputs of different release planning method when applied on the same inputs, it is sufficient that these inputs are realistic and used consistently across our experiments.

For converting BEARS probability distributions to EVOLVE estimates, we used the approach described in Section 5.1. For the Release Planner, Word Processor, and RALIC product backlogs, we first generate BEARS probability distribution as described in the previous paragraph, then generate new EVOLVE estimates that are proportional to the mean of the BEARS distributions as described in Section 5.1. When applying BEARS-*deterministic*, we use the mean of the BEARS probability distribution as single-point estimate for effort and value.

Evaluation Metrics

The first evaluation metric is to observe how often the BEARS shortlist *strictly dominates* the shortlists of other methods. In our context, a release plan $p1$ strictly dominates a release plan $p2$ if $ENPV(p1) > ENPV(p2)$ and $EP(p1) > EP(p2)$. A shortlist $L1$ strictly dominates a shortlist $L2$ if every release plan in $L2$ is strictly dominated by at least one release plan in $L1$. Strict dominance is the strongest possible form of dominance relation between sets of solutions in multi-objective optimisation problems [97]. If a single release plan in $L2$ is equal to, or is not dominated by some release plan in $L1$, then $L1$ does not strictly dominates $L2$. As in the first experiment, we have used the BEARS model (Equations 4.2 and 4.3) to compute the expected NPV and expected punctuality of release plans shortlisted by other methods.

Analysing strict dominance can tell us whether the release plans shortlisted by BEARS are better than those shortlisted by other methods but it does not tell us *how much* better. Our second evaluation metric measures the improvement of a BEARS shortlist compared to the shortlist of another method by comparing their hypervolumes. In multi-objective optimisation problems, the hypervolume of a solution set A, noted $HV(A)$ is the volume of the objective space dominated by A [98]. For example, in Figure 5.1 the hypervolume of the shortlist is the area dominated by the release plans in the shortlist, bounded below by the x-axis ($ENPV = 0$) and to the left by the y-axis ($EP = 0$). In this example, the HV of the BEARS and EVOLVE-II shortlists are 2047 and 1541, respectively. The HV of the BEARS shortlist is larger because it covers a larger area. The Hypervolume improvement ratio ($HVIR$) of a solution set A over a solution set B is $HV(A)/HV(B)$. In Figure 5.1, the HVIR of the BEARS shortlist over the EVOLVE-II shortlist is $2047/1541 = 1.33$. Hypervolume is a widely used metric to compare solution sets of multi-objective optimisation problems. We have chosen this metric because it has as simple visual interpretation and is compatible with strict dominance, i.e. if A strictly dominates B then $HV(A)/HV(B) > 1$ [97].

Experimental Set Up

The MOEAs used by the release planning methods in Table 5.1 are stochastic algorithms that can generate different results each time they are executed on a given problem. To account for such randomness, we have executed 30 independent runs of each of our 5 release planning methods on all 32 release planning problems. Each release planning method is thus executed 960 times (30×32). In total, our experiment includes $4,800$ independent runs (5×960), resulting in as many shortlists. All runs were executed on a single PC with Intel Core i5 CPU at 3.20GHz x 4 and 8GB of RAM.

5.2.2 Results

Table 5.3 reports how often BEARS generates a shortlist that strictly dominates the shortlist of other methods when each method is executed 30 times. Table 5.4 reports the mean, minimum and maximum hypervolume improvement ratios over 30 runs for each release planning problem. We have checked that all observed differences in hypervolumes between BEARS and other methods are statistically significant using Mann-Whitney U test with $p < .05$. All p-values are reported in Table 5.5.

RQ1) How does BEARS *compare to release planning methods that ignore uncertainty?* Table 5.3 shows that out of 960 runs, BEARS shortlist strictly dominates the EVOLVE-II and BEARS-*deterministic* shortlists in 96% and 97% of the runs, respectively. BEARS shortlist strictly dominates the shortlists of the two deterministic methods in all 30 runs in at least 23 of the 32 release planning problems, and strictly dominates the other shortlists in all problems in at least 21 out of 30 runs. Furthermore, Table 5.4 shows that BEARS shortlist has an hypervolume that is on average 17% and 18% higher than the EVOLVE-II and BEARS-*deterministic* shortlists, respectively. The hypervolume improvements range from 3% to 42%.

Comparing BEARS and BEARS-*deterministic* shortlists allow us to study the effect of analysing uncertainty in isolation of other factors. Tables 5.3 and 5.4 show that BEARS shortlists strictly dominates BEARS-*deterministic* shortlists and has better hypervolume improvement ratio respectively. Therefore, based on evidence from our experiment, we can conclude that analysing uncertainty during release planning leads to shortlisting different and better release plans that if uncertainty is ignored.

To help us understand the practical significance of the above results in specific

Product Backlog	H	EVOLVE-*max*	BEARS *vs.* Bears-Deterministic	Evolve-with-uncertainty	Bears-fixed-scope
Local Government Project	2	30/30	21/30	18/30	18/30
	3	30/30	30/30	19/30	21/30
	4	21/30	27/30	22/30	23/30
	5	27/30	29/30	21/30	27/30
Release Planner	2	23/30	25/30	25/30	21/30
	3	30/30	30/30	30/30	28/30
	4	30/30	30/30	30/30	30/30
	5	30/30	30/30	30/30	30/30
Word Processor	2	23/30	24/30	20/30	26/30
	3	29/30	29/30	27/30	23/30
	4	30/30	30/30	30/30	30/30
	5	30/30	27/30	26/30	30/30
RALIC	2	30/30	30/30	30/30	30/30
	3	29/30	30/30	27/30	29/30
	4	24/30	30/30	30/30	29/30
	5	22/30	30/30	20/30	27/30
Synthetic-30	2	30/30	30/30	18/30	20/30
	3	30/30	29/30	20/30	30/30
	4	30/30	30/30	21/30	30/30
	5	30/30	30/30	27/30	30/30
Synthetic-50	2	30/30	29/30	20/30	20/30
	3	30/30	30/30	20/30	21/30
	4	30/30	30/30	18/30	24/30
	5	30/30	30/30	23/30	28/30
Synthetic-100	2	30/30	30/30	21/30	28/30
	3	30/30	30/30	24/30	29/30
	4	30/30	30/30	20/30	28/30
	5	30/30	30/30	20/30	29/30
Synthetic-200	2	30/30	30/30	20/30	27/30
	3	30/30	30/30	22/30	30/30
	4	30/30	30/30	20/30	29/30
	5	30/30	30/30	25/30	30/30
Overall		918/960 (96%)	930/960 (97%)	754/960 (79%)	855/960 (89%)

Table 5.3: Proportion of runs where the BEARS shortlists strictly dominates the shortlist of other methods.

contexts, Figure 5.2 show examples of shortlists generated by BEARS and the two deterministic methods for each product backlog and a planning horizon $H = 3$. These examples all correspond to the first of the 30 runs of each method. For each release planning problem, the figure shows how much the shortlist generated by BEARS dominates the shortlists generated by the deterministic methods. In

		BEARS *vs.*							
		EVOLVE-*max*		Bears-deterministic		Evolve-with-uncertainty		Bears-fixed-scope	
Product Backlog	H	Mean	Range	Mean	Range	Mean	Range	Mean	Range
Local Government Project	2	1.22	1.19 –1.26	1.27	1.22 –1.34	1.06	1.03 –1.07	1.05	1.00 –1.12
	3	1.24	1.20 –1.33	1.30	1.22 –1.55	1.12	1.07 –1.23	1.12	1.05 –1.33
	4	1.28	1.16 –1.76	1.32	1.19 –1.51	1.11	1.07 –1.17	1.15	1.10 –1.27
	5	1.34	1.18 –1.48	1.47	1.30 –1.68	1.11	1.05 –1.19	1.16	1.11 –1.24
Release Planner	2	1.05	0.97 –1.10	1.06	1.02 –1.11	1.02	0.95 –1.07	1.01	0.93 –1.06
	3	1.07	1.05 –1.10	1.08	1.05 –1.11	1.06	1.03 –1.09	1.05	1.01 –1.08
	4	1.06	1.04 –1.09	1.06	1.04 –1.09	1.05	1.03 –1.08	1.06	1.03 –1.11
	5	1.07	1.06 –1.10	1.09	1.06 –1.15	1.06	1.03 –1.08	1.08	1.05 –1.12
Word Processor	2	1.00	0.95 –1.05	1.01	0.96 –1.05	1.00	0.94 –1.05	1.01	0.94 –1.03
	3	1.04	1.02 –1.06	1.04	1.02 –1.06	1.03	1.00 –1.05	1.03	1.00 –1.05
	4	1.06	1.05 –1.07	1.05	1.04 –1.06	1.05	1.04 –1.05	1.03	1.02 –1.04
	5	1.04	1.01 –1.04	1.03	1.00 –1.04	1.02	0.99 –1.03	1.04	1.01 –1.07
RALIC	2	1.10	1.04 –1.20	1.12	1.08 –1.16	1.07	1.03 –1.12	1.08	1.04 –1.14
	3	1.08	1.02 –1.15	1.14	1.08 –1.22	1.06	0.98 –1.12	1.08	1.02 –1.15
	4	1.09	1.02 –1.14	1.16	1.11 –1.25	1.07	1.01 –1.11	1.08	1.03 –1.15
	5	1.08	1.01 –1.15	1.16	1.07 –1.26	1.06	1.00 –1.09	1.08	1.02 –1.17
Synthetic-30	2	1.15	1.09 –1.67	1.11	1.05 –1.25	1.03	1.01 –1.09	1.02	1.00 –1.05
	3	1.14	1.09 –1.20	1.11	1.06 –1.20	1.08	1.03 –1.11	1.08	1.04 –1.13
	4	1.16	1.09 –1.23	1.15	1.10 –1.30	1.11	1.06 –1.15	1.12	1.06 –1.17
	5	1.16	1.12 –1.27	1.16	1.11 –1.22	1.14	1.11 –1.18	1.15	1.11 –1.23
Synthetic-50	2	1.15	1.07 –1.30	1.14	1.03 –1.26	1.05	1.04 –1.08	1.02	1.00 –1.06
	3	1.23	1.10 –1.36	1.25	1.12 –1.42	1.08	1.05 –1.12	1.08	1.03 –1.12
	4	1.28	1.16 –1.42	1.30	1.22 –1.45	1.13	1.05 –1.25	1.14	1.08 –1.20
	5	1.30	1.20 –1.44	1.30	1.21 –1.40	1.17	1.10 –1.26	1.20	1.14 –1.27
Synthetic-100	2	1.09	1.04 –1.14	1.08	1.05 –1.15	1.07	1.04 –1.12	1.09	1.04 –1.13
	3	1.21	1.14 –1.31	1.20	1.10 –1.33	1.15	1.09 –1.22	1.16	1.10 –1.26
	4	1.32	1.21 –1.44	1.32	1.20 –1.43	1.20	1.09 –1.32	1.22	1.12 –1.32
	5	1.39	1.28 –1.52	1.38	1.30 –1.51	1.25	1.16 –1.34	1.26	1.18 –1.35
Synthetic-200	2	1.12	1.06 –1.22	1.11	1.05 –1.20	1.15	1.07 –1.23	1.16	1.09 –1.25
	3	1.17	1.10 –1.24	1.19	1.11 –1.26	1.21	1.12 –1.29	1.24	1.13 –1.31
	4	1.28	1.21 –1.38	1.26	1.18 –1.34	1.32	1.17 –1.92	1.33	1.17 –1.55
	5	1.38	1.24 –1.50	1.40	1.23 –1.50	1.37	1.25 –1.69	1.36	1.18 –1.54
Overall		1.17	1.03 –1.39	1.18	1.03 –1.42	1.11	1.01 –1.29	1.12	1.01 –1.32

Table 5.4: Hypervolume Improvement Ratios of BEARS with respect to other release planning methods.

Figure 5.2, we observe that the dominance of BEARS over methods that ignore uncertainty is important in practice. For each of these examples, we can make similar observations to those made in our first experiment (Section 5.1).

Analysing uncertainty using BEARS leads to shortlisting release plans with higher expected NPV and expected punctuality than methods that ignore

uncertainty. In our experiments, the improvement in hypervolume range from 3% to 42%, with an average of 17%.

RQ2) How does BEARS *compare to release planing methods under uncertainty with fixed-scope releases?* Table 5.3 shows that out of 960 runs, BEARS shortlist strictly dominates the EVOLVE-*with-uncertainty* and BEARS-*fixed-scope* shortlists in 79% and 89% of the runs, respectively. BEARS shortlist strictly dominates the shortlists of EVOLVE-*with-uncertainty* and BEARS-*fixed-scope* in all 30 runs in at least 6 of the 32 release planning problems, and strictly dominates the other shortlists in all problems in at least 18 out of 30 runs. Furthermore, Table 5.4 BEARS shortlist has an hypervolume that is on average 11% and 12% higher than the EVOLVE-*with-uncertainty* and and BEARS-*fixed-scope* shortlists, respectively. The hypervolume improvements range from 1% to 32%.

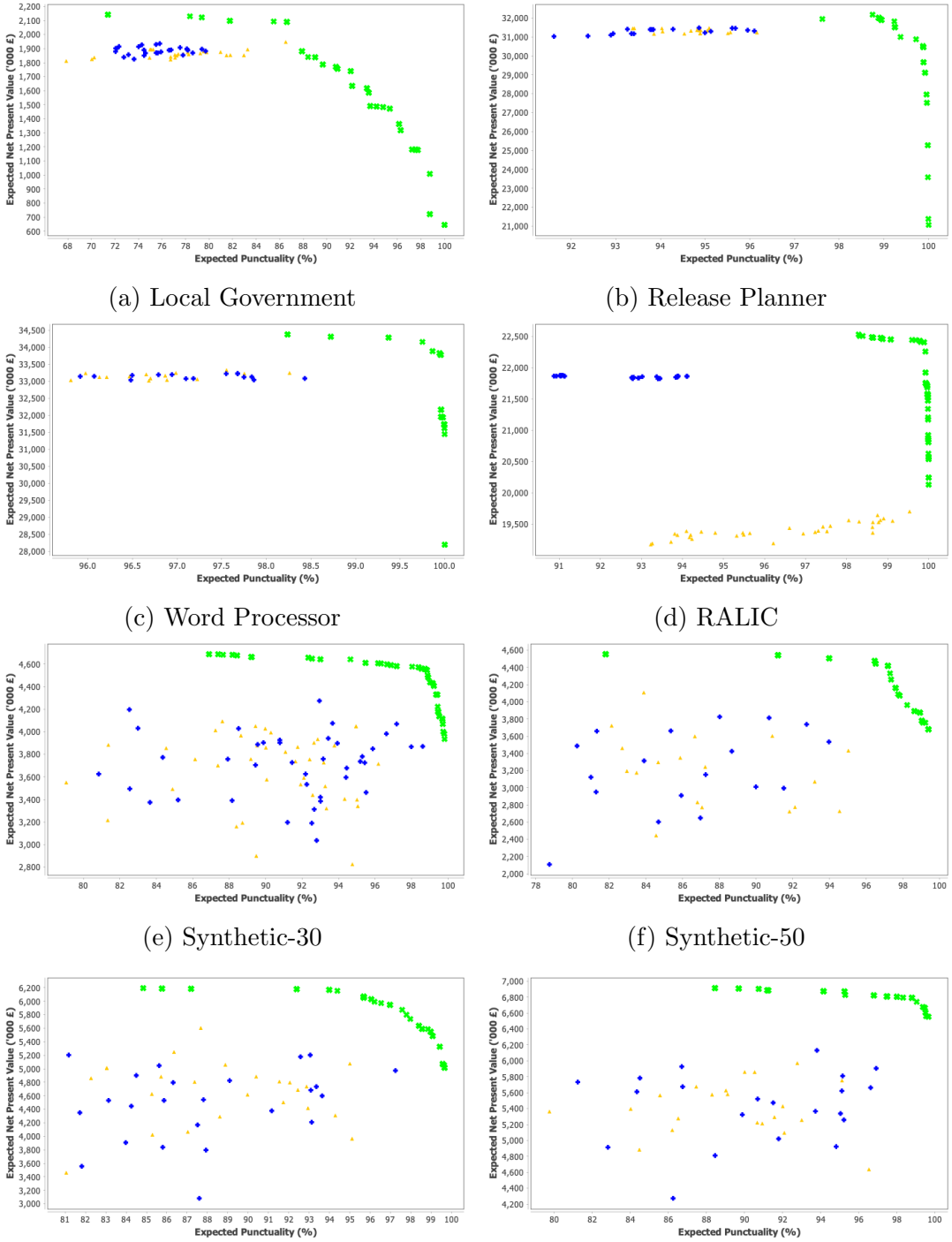
To help us understand the practical significance of the above results in specific context, Figures 5.3 show examples of shortlists generated by BEARS and analyse uncertainty assuming fixed-scope releases for each product backlog and a planning horizon $H = 3$. For each release planning problem, the figure shows how much the shortlist generated by BEARS dominates the shortlists generated by methods that analyse uncertainty assuming fixed-scope releases. In Figure 5.3, we see that the difference between BEARS and methods that analyse uncertainty assuming fixed-scope releases is smaller but still important enough to justify using BEARS for projects with fixed-date, flexible-scope release cycles.

In the context of fixed-time releases, analysing uncertainty using BEARS leads to shortlisting release plans with higher expected NPV and expected punctuality than methods designed for fixed-scope releases. In our experiments, the improvement in hypervolume range from 1% to 32%, with an average of

11%.

Product Backlog	H	EVOLVE-*max*	BEARS *vs.* Bears-deterministic	Evolve-with-uncertainty	Bears-fixed-scope
Local Government Project	2	$2.87E-11$	$2.87E-11$	$2.87E-11$	$5.77E-11$
	3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
Release Planner	2	$2.11E-07$	$2.10E-08$	$4.58E-04$	$1.25E-02$
	3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$3.51E-11$
	4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
Word Processor	2	$3.01E-01$	$6.04E-02$	$4.87E-01$	$3.85E-02$
	3	$2.87E-11$	$6.37E-11$	$1.15E-10$	$3.06E-09$
	4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	5	$2.87E-11$	$2.74E-10$	$5.84E-10$	$2.87E-11$
RALIC	2	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	3	$6.37E-11$	$2.87E-11$	$3.31E-10$	$2.05E-10$
	4	$3.18E-11$	$2.87E-11$	$3.18E-11$	$2.87E-11$
	5	$3.51E-11$	$2.87E-11$	$9.44E-11$	$3.51E-11$
Synthetic-30	2	$2.87E-11$	$2.87E-11$	$4.73E-11$	$6.41E-10$
	3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
Synthetic-50	2	$2.87E-11$	$2.87E-11$	$2.87E-11$	$7.39E-08$
	3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
Synthetic-100	2	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
Synthetic-200	2	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
	5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.87E-11$
Overall		$5.43E-116$	$3.21E-149$	$1.36E-104$	$2.71E-114$

Table 5.5: Statistical Significance (p-value) of the observed difference in hypervolume between BEARS and other methods over 30 runs using Mann-Whitney U test.



(a) Local Government

(b) Release Planner

(c) Word Processor

(d) RALIC

(e) Synthetic-30

(f) Synthetic-50

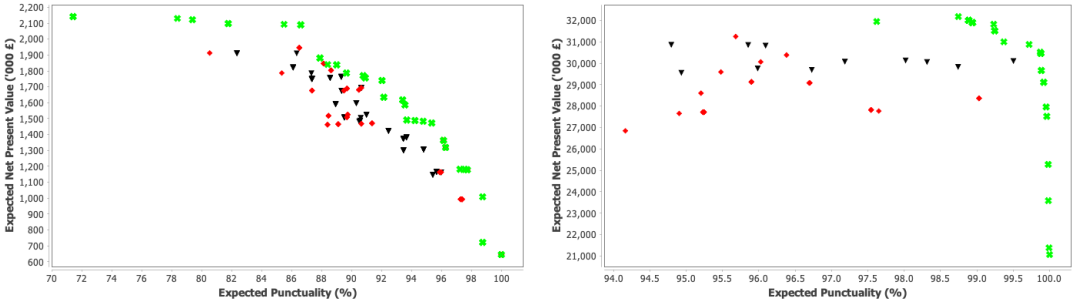
(g) Synthetic-30

(h) Synthetic-50

Figure 5.2: Examples of shortlists generated by BEARS (green crosses), EVOLVE-*max* (blue diamonds) and BEARS-deterministic (yellow triangles). These examples correspond to the first runs out of 30.

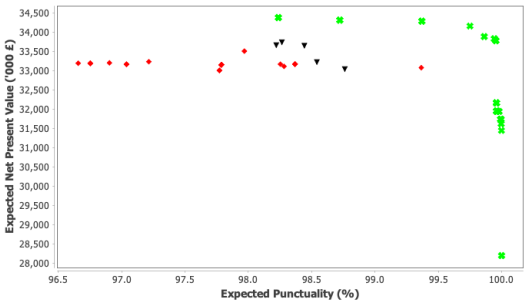
Run-times

Table 5.6 shows that BEARS is on average 25 times slower than the two methods that ignore uncertainty and 5 times slower than the two methods that analyse

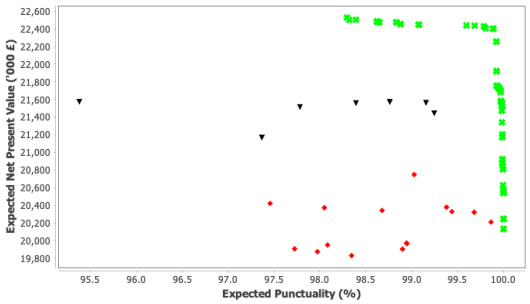


(a) Local Government

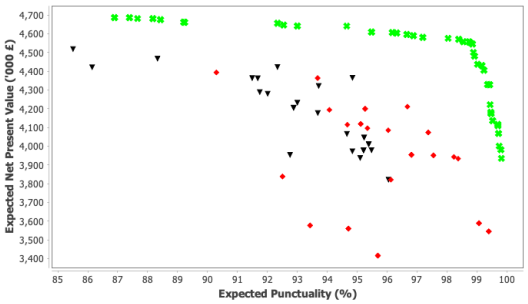
(b) Release Planner



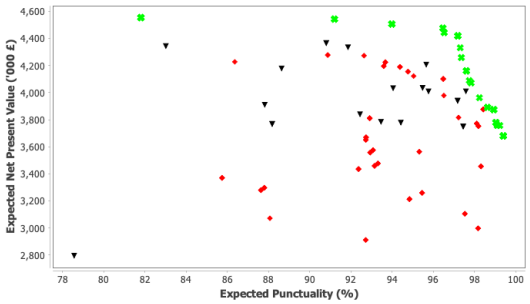
(c) Word Processor



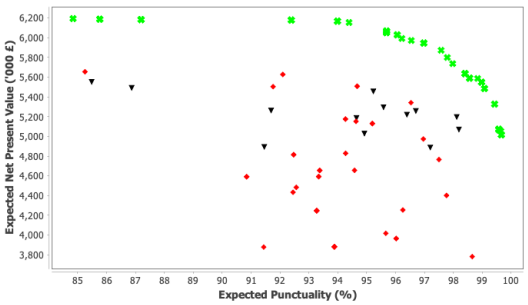
(d) RALIC



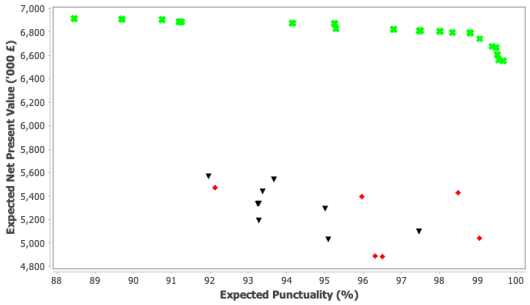
(e) Synthetic-30



(f) Synthetic-50



(g) Synthetic-30



(h) Synthetic-50

Figure 5.3: Examples of shortlists generated by BEARS (green crosses), EVOLVE-with-uncertainty (red diamonds) and BEARS-fixed-scope (black triangles). These examples correspond to the first runs out of 30.

uncertainty assuming fixed-scope releases. For our largest product backlog of 200 work items and $H = 5$, BEARS takes 10 minutes to run. Although BEARS

Product Backlog	H	EVOLVE-*max*	Bears-deterministic	Evolve-with-uncertainty	Bears-fixed-scope	BEARS
Local Government Project	2	2	3	10	18	45
	3	2	4	12	20	48
	4	3	4	15	21	60
	5	3	5	17	25	65
Release Planner	2	2	3	10	15	37
	3	2	4	14	25	55
	4	3	5	16	30	70
	5	5	8	20	35	95
Word Processor	2	2	4	18	36	95
	3	3	7	20	47	135
	4	4	8	23	58	155
	5	6	10	26	65	163
RALIC	2	4	10	37	85	153
	3	5	12	45	102	192
	4	5	12	54	123	225
	5	7	15	68	150	259
Synthetic-30	2	2	6	15	33	75
	3	3	7	17	42	95
	4	3	9	17	40	112
	5	5	11	20	49	125
Synthetic-50	2	5	9	20	68	158
	3	7	11	25	75	170
	4	7	12	29	84	188
	5	9	14	37	96	205
Synthetic-100	2	6	13	56	75	229
	3	6	16	75	94	284
	4	7	20	85	110	302
	5	7	24	97	115	357
Synthetic-200	2	9	20	83	112	288
	3	9	21	85	122	365
	4	10	22	95	130	486
	5	10	30	120	150	533

Table 5.6: Release planning methods' run-times in seconds

is significantly slower than other methods, its run time remains acceptable and does not prevent BEARS being used during release planning workshops. In such workshops, release plans are typically shortlisted only once after all work items effort and values estimates have been elicited.

5.2.3 Threats to Validity

External validity. Our results may not generalise beyond the 32 release planning problems considered in this experiment. Release planing problems where work items have different uncertainty or where other model parameters, such as team capacity, have different values could lead to different results. Further experiments on other problems are needed to refine the findings. Nevertheless, this experiment showed important differences can exist between the outputs of different release

planning methods, and we have found important differences between BEARS and methods that ignore uncertainty in all release planning problems studied so far.

Internal validity. As in the first experiment, we used the BEARS objectives of expected *NPV* and expected punctuality to evaluate release plans generated by other methods that optimise different objectives. This evaluation is appropriate because the experiments' objective is to compare different release planning methods in the context for which BEARS is designed, which is the context most commonly found in practice. Also as in the first experiment, we have deliberately aimed to minimize differences in effort and value elicitation techniques. Taking such differences into account is likely to amplify the differences found in the experiments.

Out of scope. Our experiments focused on comparing BEARS shortlists against those of other methods but have ignored other factors that are important in practice. In particular, we have not evaluated the uncertainty elicitation method used in BEARS (Section 4.2), the ability of release planners to understand BEARS' outputs, the overall cost of applying the method, and the general perceived benefits and limitations of the method by release planners, development teams and other stakeholders.

5.3 Evaluating BEARS Optimisation Algorithms

Our third experiment evaluates the performance of BEARS MOEAs. It aims to answer the following questions:

- RQ3: Do the three MOEAs used in BEARS perform better than a random search?

- RQ4: Do the three MOEAs used in BEARS have significant differences in performance?

Our objective here is not to propose novel MOEAs, nor to look for the best possible algorithms for solving BEARS optimisation problems. This experiment is a sanity check aimed at evaluating whether existing MOEAs are suitable for solving BEARS optimisation problems. Developing and evaluating more efficient algorithms for solving BEARS optimisation problems is left as future work.

5.3.1 Experiment Design

The three MOEAs used in BEARS are NSGA-II, SPEA2, and MOCell (see Section 4.4). We selected these MOEAs because they have readily available implementations in JMetal, the optimisation framework used in BEARS [89]. We compare these algorithms against a random search augmented with the same constraints violation detection and repair technique used with the MOEAs (see Section 4.4). The random search is configured to evaluate the same number of valid release plans as the MOEAs ($25,000$).

We have executed each of these 4 algorithms 30 times on the same 32 release planning problems used in our second experiment. This makes up a total of $3,840$ runs. For each run, we retrieve the generated shortlist and measure its Hypervolume (HV) [98] and modified Inverted Generational Distance (IGD+) [99].

In this experiment, we compute HV using *normalised* expected NPV values for each problem, where the minimum is 0 and maximum is the highest expected NPV found in all evaluated release plans for that problem. The HV for all problems are thus all measured on the same scale. Better shortlists have *higher* HV.

The IGD+ of a solution set A is the average distance between the true Pareto-

optimal solutions and the region in the objective space dominated by A [99]. When, as in our case, the true Pareto optimal solutions are unknown, these solutions are approximated by so-called *reference* Pareto-optimal solutions, which are the non-dominated solutions in the union of all solutions explored by all algorithms over all of independent runs. Better shortlists have *lower* IGD+.

We have chosen HV and IGD+ as our evaluation metrics based on guidance from Li et al. [100] that recommend using these metrics in situations, like ours, where the decision makers' preferences about the qualities of solutions sets are unknown. These metrics are recommended because they are compatible with the strict dominance relation, and they cover all typical qualities desired of solutions sets, i.e. their convergence (how close the solutions set is to the true Pareto front), diversity (the extent to which the solutions set includes diverse solutions), and cardinality (the number of solutions in the set).

5.3.2 Results

This section presents the conclusion of this experiment. Table 5.7 reports the median HV and IGD+ of each algorithm over 30 runs for each release planning problem. In each row, the best result is highlighted in a light grey and the worst in dark grey. We have used Mann-Whitney U test to evaluate whether the observed differences in HV and IGD+ between algorithms are statistically significant. Table 5.9 reports the results of these tests.

RQ3) Do the three MOEAs used in BEARS *perform better than a random search?* The experiment show that the three MOEAs perform better than random search and that the differences in HV and IGD+ are statistically significant. In terms of IGD+, the three MOEAS outperform random search in all 32 problems. In terms of HV, the three MOEAS outperform random search in 27 of the 32 problems.

Product Backlog	H	NSGA-II	SPEA2	MOCell	Random
Local Government Project	2	0.956	0.947	0.903	0.825
	3	0.982	0.982	0.991	0.878
	4	0.973	0.988	0.968	0.943
	5	0.975	0.982	0.996	0.888
Release Planner	2	0.806	0.807	0.851	0.752
	3	0.873	0.949	0.879	0.820
	4	0.948	0.973	0.900	0.917
	5	0.945	0.927	0.969	0.739
Word Processor	2	0.938	0.946	0.949	0.820
	3	0.971	0.988	0.999	0.907
	4	0.971	0.986	0.994	0.915
	5	0.925	0.938	0.996	0.925
RALIC	2	0.984	0.986	0.996	0.780
	3	0.967	0.992	0.994	0.686
	4	0.927	0.971	0.997	0.669
	5	0.922	0.984	0.942	0.660
Synthetic-30	2	0.594	0.598	0.517	0.526
	3	0.720	0.723	0.721	0.670
	4	0.846	0.947	0.845	0.818
	5	0.933	0.945	0.955	0.906
Synthetic-50	2	0.718	0.955	0.714	0.688
	3	0.943	0.945	0.970	0.808
	4	0.958	0.962	0.889	0.914
	5	0.963	0.968	0.965	0.841
Synthetic-100	2	0.886	0.886	0.687	0.711
	3	0.936	0.936	0.888	0.931
	4	0.948	0.948	0.948	0.758
	5	0.922	0.922	0.959	0.839
Synthetic-200	2	0.726	0.709	0.727	0.681
	3	0.975	0.945	0.961	0.956
	4	0.970	0.911	0.953	0.887
	5	0.970	0.964	0.938	0.819
Overall		0.726	0.829	0.784	0.670

Table 5.7: Mean Hypervolume Performance of a random search and the 3 Multi-Objective Optimisation Algorithms used in BEARS.

In the remaining 5 problems, random search performs better than MOCEll in 4 cases and better than SPEA2 in 1 case. Note that the random search in BEARS uses the same constraint violation detection and repair technique used by the

Product Backlog	H	NSGA-II	SPEA2	MOCell	Random
Local Government Project	2	0.019	0.011	0.009	0.080
	3	0.076	0.042	0.002	0.115
	4	0.021	0.004	0.002	0.110
	5	0.019	0.009	0.005	0.114
Release Planner	2	0.268	0.025	0.019	0.444
	3	0.424	0.088	0.020	0.635
	4	0.178	0.014	0.005	0.301
	5	0.047	0.026	0.001	0.180
Word Processor	2	0.560	0.138	0.032	1.880
	3	0.257	0.016	0.005	1.050
	4	0.076	0.035	0.001	0.525
	5	0.052	0.006	0.005	0.361
RALIC	2	0.038	0.001	0.002	6.040
	3	0.319	0.002	0.081	9.700
	4	0.455	0.005	0.008	6.260
	5	0.466	0.019	0.001	4.320
Synthetic-30	2	0.579	0.509	0.025	0.726
	3	0.383	0.251	0.378	0.412
	4	0.284	0.008	0.039	0.353
	5	0.021	0.004	0.01	0.125
Synthetic-50	2	0.216	0.011	0.161	0.216
	3	0.123	0.011	0.014	0.137
	4	0.030	0.009	0.007	0.161
	5	0.034	0.008	0.012	0.332
Synthetic-100	2	0.136	0.058	0.134	0.137
	3	0.037	0.016	0.012	0.104
	4	0.083	0.014	0.039	0.265
	5	0.083	0.025	0.024	0.506
Synthetic-200	2	0.023	0.021	0.021	0.024
	3	0.024	0.008	0.020	0.081
	4	0.061	0.003	0.048	0.274
	5	0.079	0.009	0.047	0.606
Overall		0.020	0.003	0.002	0.110

Table 5.8: Mean IGD+ Performance of a random search and the 3 Multi-Objective Optimisation Algorithms used in BEARS.

MOEAs. Random search in BEARS is therefore more than a blind search, which may explain why in a few cases it performs better than one of the three MOEA on one of the evaluation criteria, even if overall the MOEAs have better IGD+

Product Backlog	Metric	H	Random vs. NSGA-II	Random vs. SPEA2	Random vs. MOCell	NSGA-II vs. SPEA2	NSGA-II vs. MOCell	SPEA2 vs. MOCell
Local Government Project	HV	2	$2.87E-11$	$2.87E-11$	$8.68E-01$	$2.44E-03$	$2.87E-11$	$2.87E-11$
		3	$6.87E-08$	$5.45E-09$	$3.85E-08$	$4.43E-04$	$1.89E-08$	$9.76E-06$
		4	$3.88E-04$	$8.01E-06$	$5.87E-09$	$8.08E-01$	$6.11E-06$	$8.86E-04$
		5	$2.84E-05$	$5.43E-10$	$2.87E-11$	$2.83E-03$	$3.50E-08$	$5.62E-07$
	IGD+	2	$2.87E-11$	$3.96E-09$	$2.87E-11$	$9.62E-03$	$7.14E-08$	$5.70E-01$
		3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.98E-06$	$3.88E-11$	$1.43E-10$
		4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$4.09E-05$	$9.44E-11$	$2.08E-06$
		5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.47E-07$	$2.26E-10$	$1.11E-07$
Release Planner	HV	2	$3.96E-07$	$2.87E-11$	$5.32E-10$	$9.06E-01$	$4.79E-05$	$7.44E-09$
		3	$5.27E-06$	$9.44E-11$	$1.38E-05$	$3.50E-08$	$2.25E-01$	$1.47E-01$
		4	$7.10E-04$	$8.12E-09$	$2.14E-01$	$7.43E-05$	$2.19E-02$	$3.66E-07$
		5	$4.22E-05$	$4.27E-06$	$3.88E-11$	$7.01E-01$	$4.44E-02$	$1.09E-03$
	IGD+	2	$3.31E-10$	$6.81E-09$	$2.87E-11$	$1.37E-08$	$3.51E-11$	$1.65E-01$
		3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$1.63E-08$	$3.06E-09$	$1.63E-04$
		4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$4.49E-08$	$2.74E-10$	$8.71E-01$
		5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$4.99E-07$	$5.32E-10$	$5.22E-09$
Word Processor	HV	2	$1.02E-09$	$2.87E-11$	$2.87E-11$	$8.14E-03$	$1.47E-02$	$1.53E-02$
		3	$3.64E-10$	$2.87E-11$	$2.74E-10$	$7.10E-04$	$5.32E-10$	$2.98E-06$
		4	$6.07E-06$	$1.77E-09$	$9.31E-10$	$8.63E-02$	$3.21E-08$	$2.40E-06$
		5	$2.87E-01$	$4.34E-04$	$1.37E-04$	$1.17E-01$	$5.79E-05$	$2.87E-02$
	IGD+	2	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.13E-09$	$2.87E-11$	$1.12E-09$
		3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$6.81E-09$	$2.87E-11$	$1.54E-10$
		4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.71E-08$	$5.77E-11$	$7.04E-10$
		5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$3.31E-10$	$3.51E-11$	$5.10E-02$
RALIC	HV	2	$2.87E-11$	$2.87E-11$	$2.87E-11$	$7.13E-02$	$4.27E-06$	$2.82E-03$
		3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$1.80E-07$	$1.94E-09$	$4.69E-01$
		4	$7.73E-10$	$2.87E-11$	$2.87E-11$	$1.15E-06$	$2.49E-10$	$3.70E-06$
		5	$5.32E-10$	$2.87E-11$	$2.87E-11$	$1.02E-09$	$1.94E-02$	$5.32E-10$
	IGD+	2	$2.87E-11$	$2.87E-11$	$2.87E-11$	$1.35E-09$	$6.37E-11$	$9.27E-03$
		3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.49E-10$	$1.54E-10$	$7.44E-09$
		4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$1.62E-09$	$6.37E-11$	$5.65E-02$
		5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$1.62E-09$	$7.03E-11$	$9.44E-08$
Synthetic-30	HV	2	$1.17E-01$	$3.88E-04$	$7.78E-03$	$5.35E-01$	$2.76E-02$	$7.36E-02$
		3	$5.32E-10$	$1.12E-09$	$2.87E-11$	$2.14E-01$	$1.53E-02$	$8.48E-01$
		4	$6.51E-06$	$2.87E-11$	$2.87E-11$	$1.84E-04$	$1.47E-01$	$1.14E-01$
		5	$6.04E-02$	$2.82E-03$	$2.19E-04$	$3.83E-01$	$9.19E-02$	$3.59E-01$
	IGD+	2	$4.28E-07$	$2.26E-10$	$2.87E-11$	$3.21E-02$	$3.31E-10$	$1.95E-07$
		3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$1.95E-07$	$5.43E-05$	$3.26E-05$
		4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$7.73E-10$	$1.77E-08$	$1.05E-05$
		5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$5.84E-10$	$1.49E-08$	$6.98E-05$
Synthetic-50	HV	2	$5.77E-11$	$5.23E-11$	$7.49E-04$	$8.70E-08$	$1.10E-02$	$1.77E-08$
		3	$7.44E-09$	$1.37E-08$	$9.31E-10$	$2.49E-01$	$1.47E-02$	$5.10E-02$
		4	$6.16E-05$	$1.48E-09$	$2.76E-04$	$7.34E-01$	$5.70E-03$	$3.45E-06$
		5	$1.62E-09$	$2.87E-11$	$2.87E-11$	$1.60E-01$	$3.09E-02$	$6.36E-01$
	IGD+	2	$9.64E-01$	$4.40E-10$	$3.18E-11$	$3.66E-09$	$2.26E-10$	$9.44E-08$
		3	$1.11E-07$	$2.87E-11$	$2.49E-10$	$3.31E-10$	$2.68E-07$	$9.41E-01$
		4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$1.02E-09$	$3.88E-11$	$1.73E-02$
		5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.74E-10$	$1.04E-10$	$6.04E-02$
Synthetic-100	HV	2	$3.71E-05$	$3.71E-05$	$1.20E-07$	$9.99E-01$	$6.56E-05$	$6.56E-05$
		3	$6.05E-01$	$6.04E-01$	$1.02E-07$	$9.99E-01$	$3.33E-02$	$3.33E-02$
		4	$3.51E-11$	$3.51E-11$	$3.88E-11$	$9.99E-01$	$9.76E-01$	$9.76E-01$
		5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$9.99E-01$	$6.73E-04$	$6.73E-04$
	IGD+	2	$1.31E-07$	$1.31E-07$	$2.26E-10$	$9.99E-01$	$3.09E-04$	$3.09E-04$
		3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$9.99E-01$	$1.60E-01$	$1.60E-01$
		4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$9.99E-01$	$4.58E-04$	$4.58E-04$
		5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$9.99E-01$	$9.41E-01$	$9.41E-01$
Synthetic-200	HV	2	$8.12E-09$	$7.79E-03$	$2.87E-11$	$3.91E-01$	$2.37E-02$	$2.14E-01$
		3	$5.71E-09$	$1.73E-02$	$2.87E-11$	$8.01E-06$	$2.68E-05$	$2.07E-04$
		4	$4.49E-08$	$4.79E-05$	$2.87E-11$	$6.28E-07$	$1.20E-07$	$4.58E-04$
		5	$1.54E-10$	$2.87E-11$	$2.87E-11$	$7.34E-01$	$2.56E-02$	$4.10E-04$
	IGD+	2	$1.53E-02$	$2.87E-11$	$4.73E-09$	$1.54E-10$	$3.50E-08$	$2.04E-01$
		3	$2.87E-11$	$2.87E-11$	$2.87E-11$	$2.74E-10$	$1.02E-07$	$3.96E-05$
		4	$2.87E-11$	$2.87E-11$	$2.87E-11$	$7.76E-11$	$1.63E-08$	$2.10E-08$
		5	$2.87E-11$	$2.87E-11$	$2.87E-11$	$3.17E-11$	$1.23E-09$	$5.22E-09$
Overall	HV		$6.17E-98$	$2.84E-139$	$1.09E-110$	$2.71E-11$	$3.29E-06$	$3.40E-01$
	IGD+		$9.03E-110$	$1.74E-243$	$5.99E-246$	$2.30E-98$	$6.82E-105$	$9.85E-01$

Table 5.9: Statistical Significance (p-value) of the differences between MOEAs on BEARS release planning problems using Mann-Whitney U test. Highlighted cells are those where the differences are not statistically significant (p>.05)

and HV.

RQ4) Do the three MOEAs used in BEARS *have important differences in performance?* The experiment show no statistically significant differences between SPEA2 and MOCell, and a slight advantage of these two algorithms over NSGA-II. The experiment therefore suggest that SPEA2 and MOCell may find slightly better shortlists than NSGA-II for BEARS optimisation problems.

5.3.3 Threats to Validity

Our third experiment follows common guidelines and practices for evaluating the performance of stochastic multi-objective algorithms [101, 100]. This experiment is therefore subject to the usual limitations of such evaluations, notably the extent to which results can be generalized beyond the 32 release planning problems.

Chapter 6

Conclusion and Future Work

The thesis studied the following research questions:

1. How can we extend cost-value based release planning methods to analyse uncertainty?

2. How can we analyse uncertainty during release planning in the context of fixed-date release processes?

3. Does analysing uncertainty during release planning lead to better release planning decisions than if uncertainty is ignored? Would different release planning methods applied in the same context recommend the same release plans? If not, do some methods make better recommendations than others?

We addressed the first research question by proposing a release planning method called *MOIFM*. *MOIFM* extends the traditional Incremental Funding Method by analysing uncertainty and shortlisting release plans using multiple objectives. This work showed the possibility of analysing uncertainty in the context of the *IFM*. However, the release process assumed by the *IFM* is rarely used in practice. The *IFM* assumes fixed-scope release when most organisations today have release

process with fixed dates. Furthermore, the *IFM* focuses on analysing work items cost and revenue but has no explicit model of work items' development time. *MOIFM* is different from other *IFM* extensions that analyses uncertainty. Barbosa et al [18] approach allows project managers to devise a policy that maximizes the number of features built over some period of time. Alencar et al [71] approach improved algorithms for identifying optimal development sequences. Eduardo et al [75] applied game theory to analyse competitors' behaviours in a duopolistic market to maximize financial returns on software project. These extensions focused on introduced uncertainty to cash flow projections but unlike *MOIFM*, they optimise single objective.

To address the second research question, we have developed a second release planning method, called BEARS, designed for release planning decisions under uncertainty in the context of fixed-date release processes commonly used in industry. BEARS allows release planners to model uncertainty about work items development time and business value. It then uses Monte-Carlo simulation and multi-objective optimisation to shortlist release plans that maximised expected net present value and expected punctuality. Release planners can then select their preferred release plan from the shortlist based on full information about candidate release plans' expected net present value and expected punctuality. Existing release planning methods that analyses uncertainty assume fixed-scope releases [15, 28, 29, 16, 102, 27], which is inconsistent with current industrial practices. Ruhe et al [15] approach used value point to estimate value of features and analyses uncertainty about effort uncertainty using triangular distribution but ignores value uncertainty and assumes fixed-scope release. McDaid et al[16]. Lingbo et al [28] and Paixao et al [29] robust optimization approaches focuses on robustness of shortlisted solutions but assumes fixed-scope release in its simulation and not suitable for planning for multiple releases. Lingbo et al [43] proposed *METRO* framework focuses on reducing algorithmic uncertainty that originates

from simulation of uncertain effort and value parameters during optimization but *METRO* assumes fixed-scope release and does not support planning for more than one release. Pitangueira et al [27] proposed *RA-MONRP* to tackle problem faced by risk-aware stakeholders during release planning.

To study whether analysing uncertainty during release planning would lead to better decisions than if uncertainty is ignored, we have compared BEARS to alternative release planning models that ignore uncertainty on a series of example release planning problems. Our results showed that analysing uncertainty using BEARS leads to shortlisting release plans that have higher expected net present value and higher expected punctuality than if uncertainty is ignored. Previous empirical studies of release planning methods have either focused on evaluating release planning methods in industry [11, 12, 13], or on comparing the performance of alternative optimisation techniques as in our third experiment. A recent paper references 38 different evaluation of MOEAs for software release planning and presents the most comprehensive such evaluation to date [103]. Like in our third experiment, such evaluation compare the outputs of different MOEAs in the context of a single release planning method. In contrast, our first and second experiments focused on comparing alternative release planning methods. These experiments enabled us to observe important differences in the release plans shortlisted by BEARS compared to other methods.

6.1 Future Work

We highlight potential future research directions following the research presented in this thesis as follows:

6.1.1 Extend BEARS to analyse fairness and multiple value dimensions from multiple perspectives

Stakeholders concerns are not limited to maximization of financial gains. Their concerns can also include non-financial soft criteria such as customer satisfaction [25], risk factor [27], fairness analysis [30], value from multiple perspectives [30] etc. BEARS can be extended to include a third optimisation objective of maximising value points for dealing with soft-criteria in addition to its current two objectives of maximising expected net present value and expected punctuality. Soft criteria are difficult to estimate accurately because they are intangible in nature. These soft criteria can be represented using value point metrics used in previous release planning methods [6, 30].

6.1.2 Managing technical debt during release planning

BEARS can be extended to reason about technical debts in a software system as-is, model how technical debt will affect the realization of release plan objectives and determine when development effort should be allocated to fix such debts. Financial concepts that have been recently applied for managing technical debt are real options, portfolio management, cost/benefit analysis and value-based analysis [90]. Technical debts introduced into the system in previous releases can be modelled as a real option before the commencement of development for the next release. Real option will allow planners and managers to view risks and uncertainties of the technical debts in the system and exercise an option whether to fix, defer, or ignore the technical debt. The option to fix, defer or ignore will impact the value delivered over the next releases. The challenges to extend BEARS include (i) how to quantify the technical debt in the current system (ii) how to analyse the impact of each option on release objectives

6.1.3 Feedback mechanism for updating value and effort uncertainty

BEARS can be extended with feedback mechanism that allows effort and value uncertainty to be updated when new information are discovered due to change in demands, priority or market demands. Uncertainty about project parameters can be reduced when more information becomes available [41]. Current implementation of BEARS computes value of information to identify the benefits that can be realised from gathering more information about the work items but does not provide way to incorporate such information back into the model. The feedback mechanism can be modelled using statistical learning models such as Bayesian network [91].

6.1.4 Tool Evaluation of BEARS in Industrial Context

BEARS has been evaluated in this thesis using various case studies in retrospective. An important future work would be to further evaluate BEARS tool on large industrial case studies within an organisational setting. This will help in answering questions about whether applying BEARS to analyse release decisions will help planner to better manage uncertainty in the project; whether release decisions are improved compared to current state of practice in the organization; whether the decisions made from using BEARS helps to make decisions that are based on real business value. Such experiment design will be similar to a previous study carried out at Trema Laboratories, Inc [12]. The challenges of such a study are (i) finding an organisation that will allow such study in their establishment (ii) such study might require the organisation to change their process, this can create impediments as business organisations do not change their process unless there are huge business benefits.

Bibliography

[1] P. Bhawnani and G. Ruhe, "Releaseplanner-planning new releases for software maintenance and evolution." in *ICSM (Industrial and Tool Volume)*, 2005, pp. 73–76.

[2] M. Denne and J. Cleland-Huang, *Software by Numbers: Strategies for High Return, Low Risk Application Development.* Pearson Education, 2003.

[3] D. E. Morris, J. E. Oakley, and J. A. Crowe, "A web-based tool for eliciting probability distributions from experts," *Environmental modelling & software*, vol. 52, pp. 1–4, 2014.

[4] C. Larman and V. R. Basili, "Iterative and incremental developments. a brief history," *Computer*, vol. 36, no. 6, pp. 47–56, 2003.

[5] B. W. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, 1988.

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

[7] R. Günther, "Software release planning," *Handbook Software Engineering and Knowledge Engineering*, vol. 3, 2005.

[8] G. Ruhe and M. Saliu, "The art and science of software release planning," *IEEE Software*, vol. 22, no. 6, pp. 47–53, 2005.

[9] M. Svahnberg, T. Gorschek, R. Feldt, R. Torkar, S. B. Saleem, and M. U. Shafique, "A systematic review on strategic release planning models," *Information and software technology*, vol. 52, no. 3, pp. 237–248, 2010.

[10] D. Ameller, C. Farré, X. Franch, and G. Rufian, "A survey on software release planning models," in *Product-Focused Software Process Improvement: 17th International Conference, PROFES 2016, Trondheim, Norway, November 22-24, 2016, Proceedings 17*. Springer, 2016, pp. 48–65.

[11] Amandeep, G. Ruhe, and M. Stanford, *Intelligent Support for Software Release Planning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 248–262.

[12] J. Momoh and G. Ruhe, "Release planning process improvement—an industrial case study," *Software Process: Improvement and Practice*, vol. 11, no. 3, pp. 295–307, 2006.

[13] M. Lindgren, R. Land, C. Norström, and A. Wall, "Key aspects of software release planning in industry," in *19th Australian Conference on Software Engineering (aswec 2008)*. IEEE, 2008, pp. 320–329.

[14] S. A. Busari and E. Letier, "Radar: A lightweight tool for requirements and architecture decision analysis," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 2017, pp. 552–562.

[15] G. Ruhe and D. Greer, "Quantitative studies in software release planning under risk and resource constraints," in *Proceedings of the 2003 International Symposium on Empirical Software Engineering*, ser. ISESE '03. Washington, DC, USA: IEEE Computer Society, 2003, p. 262.

[16] K. McDaid, D. Greer, F. Keenan, P. Prior, G. Coleman, and P. S. Taylor, "Managing uncertainty in agile release planning." in *SEKE*, 2006, pp. 138–143.

[17] K. Logue and K. McDaid, "Agile release planning: Dealing with uncertainty in development time and business value," in *Engineering of Computer Based Systems, 2008. ECBS 2008. 15th Annual IEEE International Conference and Workshop on the.* IEEE, 2008, pp. 437–442.

[18] B. P. Barbosa, E. A. Schmitz, and A. J. Alencar, "Generating software-project investment policies in an uncertain environment," in *Systems and Information Engineering Design Symposium, 2008. SIEDS 2008. IEEE.* IEEE, 2008, pp. 178–183.

[19] D. Leffingwell, *Agile software requirements: lean requirements practices for teams, programs, and the enterprise.* Addison-Wesley Professional, 2010.

[20] A. J. Bagnall, V. J. Rayward-Smith, and I. M. Whittley, "The next release problem," *Information and software technology*, vol. 43, no. 14, pp. 883–890, 2001.

[21] D. Greer and G. Ruhe, "Software release planning: an evolutionary and iterative approach," *Information and software technology*, vol. 46, no. 4, pp. 243–253, 2004.

[22] M. Denne and J. Cleland-Huang, "The incremental funding method: Data-driven software development," *IEEE Software*, vol. 21, no. 3, pp. 39–47, 2004.

[23] A. Shalloway, G. Beaver, and J. R. Trott, *Lean-agile software development: achieving enterprise agility.* Pearson Education, 2009.

[24] Y. Zhang, M. Harman, and S. L. Lim, "Empirical evaluation of search based requirements interaction management," *Information and Software Technology*, vol. 55, no. 1, pp. 126–152, 2013.

[25] Y. Zhang, M. Harman, and S. A. Mansouri, "The multi-objective next release problem," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. ACM, 2007, pp. 1129–1137.

[26] Y. Zhang, E. Alba, J. J. Durillo, S. Eldh, and M. Harman, "Today/future importance analysis," in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM, 2010, pp. 1357–1364.

[27] A. M. Pitangueira, P. Tonella, A. Susi, R. S. Maciel, and M. Barros, *Requirements Engineering: Foundation for Software Quality: 22nd International Working Conference, REFSQ 2016, Gothenburg, Sweden, March 14-17, 2016, Proceedings*. Cham: Springer International Publishing, 2016, ch. Risk-Aware Multi-stakeholder Next Release Planning Using Multi-objective Optimization, pp. 3–18.

[28] L. Li, M. Harman, E. Letier, and Y. Zhang, "Robust next release problem: handling uncertainty during optimization," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2014, pp. 1247–1254.

[29] M. Paixao and J. Souza, "A robust optimization approach to the next release problem in the presence of uncertainties," *J. Syst. Softw.*, vol. 103, no. C, pp. 281–295, May 2015.

[30] A. Finkelstein, M. Harman, S. A. Mansouri, J. Ren, and Y. Zhang, ""fairness analysis" in requirements assignments," in *International Requirements Engineering, 2008. RE'08. 16th IEEE*. IEEE, 2008, pp. 115–124.

[31] M. Cohn, *Agile estimating and planning*. Pearson Education, 2005.

[32] C. K.Beck, "Extreme Programming Explained," *Writing*, pp. 85–110, 2005.

[33] M. Jorgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Transactions on software engineering*, vol. 33, no. 1, pp. 33–53, 2007.

[34] M. Jørgensen, "A review of studies on expert estimation of software development effort," *Journal of Systems and Software*, vol. 70, no. 1, pp. 37–60, 2004.

[35] B. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy, and R. Selby, "Cost models for future software life cycle processes: Cocomo 2.0," *Annals of software engineering*, vol. 1, no. 1, pp. 57–94, 1995.

[36] J. Grenning, "Planning poker," *Renaissance Software Consulting*, 2002.

[37] N. C. Haugen, "An empirical study of using planning poker for user story estimation," in *AGILE 2006 (AGILE'06)*. IEEE, 2006, pp. 9–pp.

[38] M. Jorgensen, "What we do and don't know about software development effort estimation," *IEEE software*, vol. 31, no. 2, pp. 37–40, 2014.

[39] S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, and P. Grünbacher, *Value-based software engineering*. Springer Science & Business Media, 2006.

[40] B. W. Boehm, "Value-based software engineering: Overview and agenda," in *Value-based software engineering*. Springer, 2006, pp. 3–14.

[41] E. Letier, D. Stefan, and E. T. Barr, "Uncertainty, risk, and information value in software requirements and architecture," in *36th International Conference on Software Engineering (ICSE 2014)*, 2014, pp. 883–894.

[42] M. Khurum, T. Gorschek, and M. Wilson, "The software value map—an exhaustive collection of value aspects for the development of software intensive products," *Journal of Software: Evolution and Process*, vol. 25, no. 7, pp. 711–741, 2013.

[43] L. Li, M. Harman, F. Wu, and Y. Zhang, "The value of exact analysis in requirements selection," 2016.

[44] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing requirements," *IEEE software*, vol. 14, no. 5, pp. 67–74, 1997.

[45] L. Lehtola, M. Kauppinen, and S. Kujala, "Requirements prioritization challenges in practice," in *International Conference on Product Focused Software Process Improvement*. Springer, 2004, pp. 497–508.

[46] P. Carlshamre, "Release Planning in Market-Driven Software Product Development: Provoking an Understanding," *Requirements Engineering*, vol. 7, pp. 139–151, 2002.

[47] B. Regnell, P. Beremark, and O. Eklundh, "A market-driven requirements engineering process: results from an industrial process improvement programme," *Requirements Engineering*, vol. 3, no. 2, pp. 121–129, 1998.

[48] B. Flyvbjerg, "Policy and Planning for Large Infrastructure Projects," no. December, p. 32, 2005.

[49] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *Trans. Evol. Comp*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[50] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," 2001.

[51] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjective optimization," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*. Ieee, 1994, pp. 82–87.

[52] Y. Zhang and M. Harman, "Search based optimization of requirements interaction management," in *Search Based Software Engineering (SSBSE), 2010 Second International Symposium on*. IEEE, 2010, pp. 47–56.

[53] M. O. Saliu and G. Ruhe, "Bi-objective release planning for evolving software systems," in *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*. ACM, 2007, pp. 105–114.

[54] N. Agarwal, R. Karimpour, and G. Ruhe, "Theme-based product release planning: An analytical approach," in *System Sciences (HICSS), 2014 47th Hawaii International Conference on*. IEEE, 2014, pp. 4739–4748.

[55] M. R. Karim and G. Ruhe, "Bi-objective genetic search for release planning in support of themes," in *International Symposium on Search Based Software Engineering*. Springer, 2014, pp. 123–137.

[56] O. Saliu and G. Ruhe, "Supporting software release planning decisions for evolving systems," in *29th Annual IEEE/NASA Software Engineering Workshop*. IEEE, 2005, pp. 14–26.

[57] M. Li, M. Huang, F. Shu, and J. Li, "A risk-driven method for extreme programming release planning," in *Proceedings of the 28th international conference on Software engineering*. ACM, 2006, pp. 423–430.

[58] D. Pfahl, A. Al-Emran, and G. Ruhe, "A system dynamics simulation model for analyzing the stability of software release plans," *Software Process: Improvement and Practice*, vol. 12, no. 5, pp. 475–490, 2007.

[59] A. Al-Emran and D. Pfahl, "Operational planning, re-planning and risk analysis for software releases," in *International Conference on Product Focused Software Process Improvement*. Springer, 2007, pp. 315–329.

[60] A. Ngo-The and M. O. Saliu, "Fuzzy structural dependency constraints in software release planning," in *The 14th IEEE International Conference on Fuzzy Systems, 2005. FUZZ'05.* IEEE, 2005, pp. 442–447.

[61] A. Ngo-The and O. Saliu, "Measuring dependency constraint satisfaction in software release planning using dissimilarity of fuzzy graphs," in *Fourth IEEE Conference on Cognitive Informatics, 2005.(ICCI 2005).* IEEE, 2005, pp. 301–307.

[62] G. Ruhe and A. Ngo, "Hybrid intelligence in software release planning," *Int. J. Hybrid Intell. Syst.*, vol. 1, no. 1-2, pp. 99–110, Apr. 2004.

[63] S. Maurice, G. Ruhe, O. Saliu *et al.*, "Decision support for value-based software release planning," in *Value-Based Software Engineering.* Springer, 2006, pp. 247–261.

[64] T. AlBourae, G. Ruhe, and M. Moussavi, "Lightweight replanning of software product releases," in *2006 International Workshop on Software Product Management (IWSPM'06-RE'06 Workshop).* IEEE, 2006, pp. 27–34.

[65] L. Karlsson, "Requirements prioritisation and retrospective analysis for release planning process improvement," Ph.D. dissertation, Lund University, 2006.

[66] B. Regnell and K. Kuchcinski, "Exploring software product management decision problems with constraint solving-opportunities for prioritization and release planning," in *Software Product Management (IWSPM), 2011 Fifth International Workshop on.* IEEE, 2011, pp. 47–56.

[67] A. Finkelstein, M. Harman, S. A. Mansouri, J. Ren, and Y. Zhang, "A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making," *Requirements Engineering*, vol. 14, no. 4, pp. 231–245, 2009.

[68] D. Hubbard, *How to Measure Anything: Finding the Value of Intangibles in Business*. Wiley, 2010.

[69] S. Tockey, *Return on software*. Addison-Wesley, 2005.

[70] M. Cantor, "Calculating and improving roi in software and system programs," *Commun. ACM*, vol. 54, no. 9, pp. 121–130, Sep. 2011.

[71] A. J. Alencar, C. A. Franco, E. A. Schmitz, and A. L. Correa, "A statistical approach for the maximization of the financial benefits yielded by a large set of mmfs and aes," *Computing and Informatics*, vol. 32, no. 6, pp. 1147–1169, 2014.

[72] A. O'Hagan, C. Buck, A. Daneshkhah, J. Eiser, P. Garthwaite, D. Jenkinson, J. Oakley, and T. Rakow, *Uncertain Judgements: Eliciting Experts' Probabilities*. Wiley, 2006.

[73] Hubbard, "Applied Information Economics : A New Method for Quantifying IT Value An Executive Overview," *Decision Analysis*, p. 6, 2004.

[74] E. Mattos, M. Vieira, E. A. Schmitz, and A. J. Alencar, "Applying game theory to the incremental funding method in software projects," *Journal of Software*, vol. 9, no. 6, pp. 14–35, 2014.

[75] E. M. Da Cunha Mattos, M. Vieira, E. A. Schmitz, and A. J. Alencar, "Applying Game Theory to the Incremental Funding Method in Software Projects," *Journal of Software*, vol. 9, no. 6, pp. 1435–1443, 2014.

[76] J. Oakley and A. O'Hagan, "Shelf: the sheffield elicitation framework (version 3.0)," in *SHELF*. School of Mathematics and Statistics, University of Sheffield, UK. (http://tonyhagan.co.uk/shelf), 2016.

[77] B. Flyvbjerg and A. Budzier, "Why Your IT Project You Think," *Harvard Business Review*, no. September 2011, 2011.

[78] S. Group *et al.*, "Chaos summary 2009," *Online report. Accessed June*, vol. 20, 2009.

[79] S. Group, "Standish chaos report," *Chaos report*, 2014.

[80] T. DeMarco and T. Lister, *Waltzing with Bears: Managing Risk on Software Projects*. New York, NY, USA: Dorset House Publishing Co., Inc., 2003.

[81] R. Darimont and A. Van Lamsweerde, "Formal refinement patterns for goal-driven requirements elaboration," in *ACM SIGSOFT Software Engineering Notes*, vol. 21, no. 6. ACM, 1996, pp. 179–190.

[82] E. Letier and A. Van Lamsweerde, "Reasoning about partial goal satisfaction for requirements and design engineering," in *ACM SIGSOFT Software Engineering Notes*, vol. 29, no. 6. ACM, 2004, pp. 53–62.

[83] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*, 1st ed. Wiley Publishing, 2009.

[84] A. O'Hagan and J. E. Oakley, "Probability is perfect, but we can't elicit it perfectly," *Reliability Engineering & System Safety*, vol. 85, no. 1, pp. 239–248, 2004.

[85] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba, "Mocell: A cellular genetic algorithm for multiobjective optimization," *Int. J. Intell. Syst.*, vol. 24, no. 7, pp. 726–746, Jul. 2009.

[86] J. Sagrado, I. M. Águila, and F. J. Orellana, "Multi-objective ant colony optimization for requirements selection," *Empirical Softw. Engg.*, vol. 20, no. 3, pp. 577–610, Jun. 2015.

[87] J. C. Felli and G. B. Hazen, "Sensitivity analysis and the expected value of perfect information," *Medical Decision Making*, vol. 18, no. 1, pp. 95–109, 1998.

[88] M. Sadatsafavi, N. Bansback, Z. Zafari, M. Najafzadeh, and C. Marra, "Need for speed: an efficient algorithm for calculation of single-parameter expected value of partial perfect information," *Value in Health*, vol. 16, no. 2, pp. 438–448, 2013.

[89] A. J. Nebro, J. J. Durillo, and M. Vergne, "Redesigning the jmetal multi-objective optimization framework," in *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO Companion '15. New York, NY, USA: ACM, 2015, pp. 1093–1100.

[90] A. Ampatzoglou, A. Ampatzoglou, A. Chatzigeorgiou, and P. Avgeriou, "The financial aspect of managing technical debt: A systematic literature review," *Information and Software Technology*, vol. 64, pp. 52–73, 2015.

[91] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe, "Toward data-driven requirements engineering," *IEEE Software*, vol. 33, no. 1, pp. 48–54, 2016.

[92] G. Ruhe *et al.*, "A systematic approach for solving the wicked problem of software release planning," *Soft Computing*, vol. 12, no. 1, pp. 95–108, 2008.

[93] A. A. Araújo, M. Paixao, I. Yeltsin, A. Dantas, and J. Souza, "An architecture based on interactive optimization and machine learning applied to the next release problem," *Automated Software Engineering*, vol. 24, no. 3, pp. 623–671, 2017.

[94] Y. Zhang, M. Harman, G. Ochoa, G. Ruhe, and S. Brinkkemper, "An empirical study of meta-and hyper-heuristic search for multi-objective release planning," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 27, no. 1, p. 3, 2018.

[95] S. L. Lim and A. Finkelstein, "StakeRare: using social networks and collaborative filtering for large-scale requirements elicitation," *IEEE transactions on software engineering*, vol. 38, no. 3, pp. 707–735, 2012.

[96] N. A. Office, "Over-optimism in government projects," 2013.

[97] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on evolutionary computation*, vol. 7, no. 2, pp. 117–132, 2003.

[98] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, pp. 257–271, 1999.

[99] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, "Modified distance calculation in generational distance and inverted generational distance," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2015, pp. 110–125.

[100] M. Li, T. Chen, and X. Yao, "A critical review of: a practical guide to select quality indicators for assessing pareto-based search algorithms in search-based software engineering: essay on quality indicator selection for SBSE," in *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*. ACM, 2018, pp. 17–20.

[101] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11. New York, NY, USA: ACM, 2011, pp. 1–10.

[102] K. Logue and K. McDaid, "Agile release planning: Dealing with uncertainty in development time and business value," *Proceedings - Fifteenth IEEE*

International Conference and Workshops on the Engineering of Computer-Based Systems, ECBS 2008, pp. 437–442, 2008.

[103] Y. Zhang, M. Harman, G. Ochoa, G. Ruhe, and S. Brinkkemper, "An empirical study of meta-and hyper-heuristic search for multi-objective release planning," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 27, no. 1, p. 3, 2018.