

Quantitative release planning in Extreme Programming

Gert van Valkenhoef^{*a,b}, Tommi Tervonen^c, Bert de Brock^b, Douwe Postmus^a

^a*Department of Epidemiology, University Medical Center Groningen, The Netherlands*

^b*Faculty of Economics and Business, University of Groningen, The Netherlands*

^c*Econometric Institute, Erasmus University Rotterdam, The Netherlands*

Abstract

Context: Extreme Programming (XP) is one of the most popular agile software development methodologies. XP is defined as a consistent set of values and practices designed to work well together, but lacks practices for project management and especially for supporting the customer role. The customer representative is constantly under pressure and may experience difficulties in foreseeing the adequacy of a release plan.

Objective: To assist release planning in XP by structuring the planning problem and providing an optimization model that suggests a suitable release plan.

Method: We develop an optimization model that generates a release plan taking into account story size, business value, possible precedence relations, themes, and uncertainty in velocity prediction. The running-time feasibility is established through computational tests. In addition, we provide a practical heuristic approach to velocity estimation.

Results: Computational tests show that problems with up to 6 themes and 50 stories can be solved exactly. An example provides insight into uncertainties affecting velocity, and indicates that the model can be applied in practice.

Conclusion: An optimization model can be used in practice to enable the customer representative to take more informed decisions faster. This can help adopting XP in projects where plan-driven approaches have traditionally been used.

Key words: Extreme programming, project management, customer role, integer programming

1. Introduction

Extreme Programming (XP) is an agile software development methodology. XP defines software development through values and practices thought to work well together in practice [1, 2]. In XP, good project management and strong customer involvement are critical for project success [3]. Yet, XP provides very little project management support [4] and the XP customer is consistently under significantly more pressure than the developers or other participants in the project [5]. Release planning in particular has been characterized as a difficult problem in which many variables have to be considered and judgments are primarily relative [6], leading to user story overload: it is unpractical to consider everything, even with a moderate number of stories. In addition, the customer may suffer from prioritization stress: it is difficult to foresee the consequences and adequacy of prioritization [5], and it is unclear if the customer perceives business value in constantly managing the development priorities [7]. Therefore, tool

support for exploring the solution space and for generating potential solutions is desired [6]. In the distinction between art and science in release planning [8], planning in XP is traditionally all art [9]. Easy to use low effort planning software could enable a hybrid approach, but would need to be tailored for the XP practices and values [2].

Quantitative models for supporting software release planning have been proposed previously, but a recent systematic review concludes that there is a lack of diversity among the existing models [10]. Quantitative approaches for requirements prioritization based on value and effort in a plan-driven context were developed in [6, 11], and a general mathematical formulation for incremental development was proposed in [8]. More fine-grained models that take into account resource constraints due to developers with varying capabilities have been proposed for iterative development [12, 13, 14]. Several approaches handle uncertainty in the parameters by generating multiple ‘good’ plans [8, 12, 15, 16, 17] rather than a single optimal one. Another model [18] can be used together with Scrum to deal with change, but it does not explicitly consider uncertainty during planning.

If a high degree of requirements change is expected, agile methods are better suited than plan-driven ones and consequently it may be necessary to adopt agile methods outside of their home ground [19]. To enable this, our previous work introduced additional product and re-

*Corresponding author, Department of Epidemiology, University Medical Center Groningen, PO Box 30.001, 9700 RB Groningen, The Netherlands. Tel.: +31 50 361 4522.

Email addresses: g.h.m.van.valkenhoef@rug.nl (Gert van Valkenhoef), tervonen@ese.eur.nl (Tommi Tervonen), e.o.de.brock@rug.nl (Bert de Brock), d.postmus@epi.umcg.nl (Douwe Postmus)

lease planning practices for XP [20]. One of the introduced practices uses an optimization model based on evaluating user stories not only on their sizes (i.e. implementation effort), but also based on business value (on an interval scale), possible precedence relations (i.e., story x needs to be completed before story y), and themes. In contrast to the models described previously, our model is tailored to XP as it embraces change by generating only a high-level global plan in terms of user stories, leaving room for agility in how these are realized, and which stories are left out when things do not go according to the plan. This paper extends the model proposed in [20] to take into account uncertain project velocity.

There are few models that plan for uncertainty explicitly. In [21], a simulation-based approach is introduced for evaluating the impact of uncertainty on the execution of release plans, but it does not enable generating a plan. Only one of the models in the literature assumes a probability distribution for the effort estimates and can generate plans with pre-specified completion probabilities [15]. However, reality can (and will) deviate from the plan depending on many other factors, and Bayesian models can aggregate these uncertainties in the probability density of the velocity [22]. In our model parameter uncertainty is accounted for by assuming a probability distribution for the velocity.

The remainder of this paper is structured as follows. We start with a general overview of the model in Section 2. Then, in Section 3, we give the integer programming formulation of our quantitative planning model. Section 4 addresses the problem of theme value elicitation and Section 5 introduces a simple method to estimate a probability distribution for the velocity. We evaluate the velocity estimation heuristic with an example in Section 6. Section 7 evaluates the running time of the model. Finally, we conclude and give directions for future work in Section 8.

2. Release planning model

We present a model to assist the XP development team, especially the customer, in the release planning process. First we briefly explain the context of planning in XP and clarify the used terminology. Then we provide a global overview of the model structure including the required inputs and produced outputs. Finally, we discuss how the model fits within the XP development process and the potential caveats in its application.

2.1. Planning in XP

The central idea of planning in XP is to plan features to implement rather than the development tasks necessary to implement these features. Planning features, represented by user stories, enables measuring progress through verifiable functionality. When the number of stories to implement is large, functionally related stories can be grouped into themes that form a consistent set of desirable features

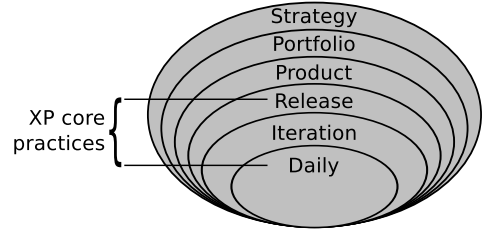


Figure 1: A Planning Onion [23] showing the levels of granularity in XP planning. XP core practices consider daily, iteration, and release planning.

[23]. This enables planning to take place at a higher level of abstraction before considering the merits of individual stories. To deal with uncertainty, the planning process occurs iteratively at release, iteration, and daily levels (see Figure 1). In *release planning*, the user stories (themes) to be developed for the next release are chosen. The developers coarsely estimate (in story points) the implementation effort required for each story and the customer prioritizes the stories. Together they agree on a high level plan consisting of critical stories that are likely to be implemented during the release, and non-critical stories that may or may not be implemented. The release cycle is fixed at 3–6 months [23] and is made up of 1–4 week iterations [1, 2, 9, 23] that each result in a working system. In *iteration planning*, the development team breaks stories with high priority down into tasks and estimates the effort required to implement these tasks in order to decide which stories can be implemented during the iteration. This also results in an updated release plan. Work division and task scheduling takes place only at the *daily planning* level.

The amount of implementation effort (story points) available during a release depends on the length of the release cycle, and the *velocity* of the development team. The velocity is the number of story points that can be implemented during some unit of time (e.g. an iteration). Traditionally the velocity is predicted through simple methods [9], and uncertainty assessed using rules of thumb [23]. However, there also exists a dynamic Bayesian network model for project velocity monitoring and prediction in XP projects that explicitly quantifies uncertainty [22].

2.2. Model overview

Our model aims at supporting release planning by generating a release plan that maximizes the implemented business value taking into account capacity constraints, precedence relations, themes, and uncertainty in the velocity. Applying the model results in a suggested release plan, which consists of sets of stories ordered according to decreasing completion probability. During release planning, the team may decide to accept the suggested plan as-is, or to amend it in any way they see fit. The release plan serves as the main input for iteration planning.

2.2.1. Model inputs

Our model requires explicit elicitation of a number of parameters (responsible team member in parentheses):

- story and theme values (customer)
- story sizes (developers)
- preference precedences (customer)
- technical precedences (developers)
- a velocity prediction (tracker)

Story size and value elicitation should be done by analogy, so that a story’s size and value is defined relative to the size and value of other (past and present) stories. We propose using a 1-5 scale for story values as most customers are already familiar with it due to its similarity with the Likert scale that is widely applied in questionnaires. If large differences in story values (i.e. more than a factor 5) exist, a wider scale needs to be used. In some projects a metric of predicted monetary value [24] may be more suitable. Synergy effects should occur when all stories within a theme are implemented. Similar to how [13] implemented revenue-based dependencies, we model such effects by awarding extra value to a theme of stories when all stories are implemented. How this extra value can be specified is discussed in Section 4. Story size is estimated in story points, for which e.g. the Cohn scale [23] can be used.

Two types of precedence relations exist: technical and preference. Technical precedence means that a story cannot technically be implemented before another one. Although stories in XP should be as independent as possible [9], sometimes dependencies are unavoidable [6]. Preference precedence allows the client to express preferences for stories, and is interpreted as a story not having value unless another (preceding) story is implemented.

Imprecision in the estimated story sizes, variability in programmer productivity, and uncertainty in other factors mean that release velocity is uncertain. This is accounted for in our model by assuming a probability distribution $f(v)$ for the total number of story points that can be implemented during the release. Our method can be used with any method that estimates $f(v)$. The construction of $f(v)$ conditional on knowledge about the underlying factors is discussed further in [22]. We propose a simple heuristic estimation procedure for situations where this approach is too demanding (Section 5).

2.2.2. Model outputs

Due to uncertainty in velocity, the optimal planning decision is a stochastic problem. In order to provide simple rules for release planning in practice, our model gives (disjoint) sets of stories with decreasing completion probabilities. Let $p_k \in (0, 1)$ denote the completion probability

of story set ℓ_k . Then, the story sets are ordered in such a way that

$$p_i > p_j ; \forall_{i < j}$$

Normally it is sufficient to distinguish three story sets corresponding to the Dynamic Systems Development Method MoSCoW rules [23, 25]: ‘must have’ (ℓ_1), ‘should have’ (ℓ_2), and ‘could have’ (ℓ_3). For example, we could set $p_1 = 0.9$, $p_2 = 0.7$, and $p_3 = 0.3$ as the desired completion probabilities for (respectively) the ‘must have’, ‘should have’, and ‘could have’ sets. Optionally a ‘won’t have’ set can be used to store the rest of the story backlog. A similar approach is taken in [26], where it is proposed to plan time-bound projects in several increments with decreasing completion probabilities. However, in our model, the story sets do not correspond directly to iterations as the scope of an iteration is decided in iteration planning.

2.3. Discussion

The model we propose fits well in the XP development process, as it assists in release planning while leaving the other XP practices intact. Moreover, we minimize the data elements that have to be specified. Compared to standard XP planning, we additionally elicit the value of stories and themes, which is already implicit in the current process because the customer prioritizes the stories. For velocity we use a probability distribution rather than just a point estimate, but the required calculations are based on historical data that is already available (see Section 5). The role of the model within the XP planning cycle is illustrated in Figure 2.

The model is not intended for planning multiple releases, as doing so might lower the overall agility of the XP development process. However, in some situations it may be crucial for the project success to plan multiple releases ahead, for example due to marketing reasons. Such long term plans can be handled otherwise, e.g. with the rolling forecast practice [20]. In addition, our model assumes adherence to the standard best practices regarding story and theme sizes [23]. Stories should be small enough so that they can easily be implemented in a single iteration. Themes should be small enough to be implemented in a single release. Moreover, the customer should understand that a theme should consist of the *minimal set of stories* required to achieve the aforementioned synergy effect, analogous to the concept of minimum marketable features [27, 28]. This ensures that the development team does not overcommit itself to a specific theme, thereby missing an opportunity to create more value elsewhere.

In iteration planning, the ‘must have’ stories are considered first. Since the completion probabilities $p_k < 1$, it is to be expected that not all planned stories will be implemented. Stories that are unlikely to be completed in the current release will be considered for inclusion in the next one, which is planned near the end of the current one.

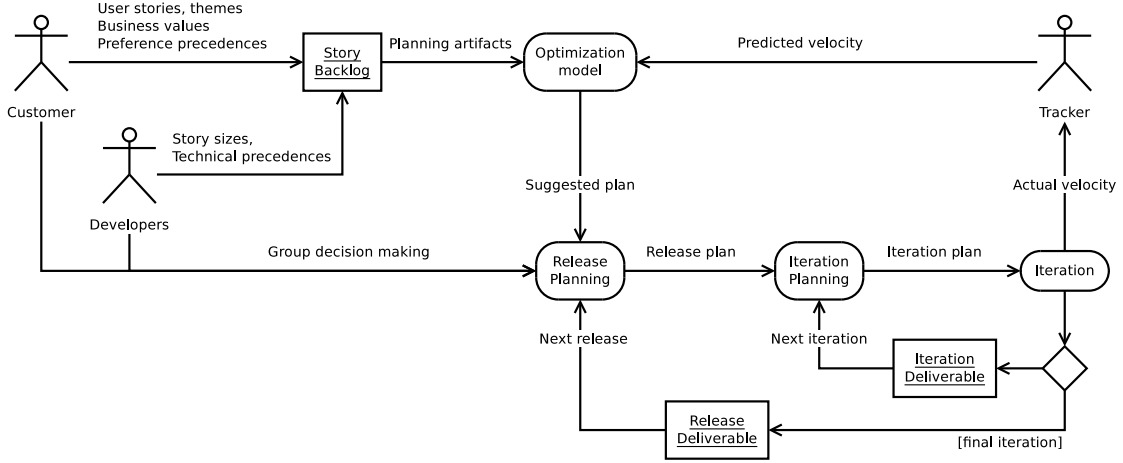


Figure 2: The role of our proposed optimization model in the XP planning process. Release and iteration planning are shown and the other levels of planning are omitted.

3. Nested knapsack formulation

Our planning model is an instance of a nested knapsack problem [29]. It is defined by having a set of “knapsacks”, each with a limited capacity, and a set of items, each with a size and value. The knapsacks themselves are nested, meaning that they are ordered in such a way that each knapsack contains the preceding one. The problem is then to maximize the value that fits into these knapsacks without exceeding their size limits. The problem is known to be NP-complete [29].

Let us define an index set of stories $S = \{1, \dots, n\}$ and of themes $T = \{n+1, \dots, n+m\}$. All stories s_i and themes t_j have a business value, respectively u_i and u_j , and stories additionally have a size c_i :

$$\begin{aligned} u_i &\in \mathbb{N}; i \in S \cup T \\ c_i &\in \mathbb{N}; i \in S \end{aligned}$$

Let $L = \{1, \dots, q\}$ denote the index set of story sets. Define q nested knapsacks, $\kappa_1, \dots, \kappa_q$, such that

$$\kappa_k = \cup_{i \leq k} \ell_i; k \in L$$

In contrast to the story sets, which are disjoint, each knapsack contains all preceding knapsacks. Associated with each knapsack κ_k is a budget $b_k \in \mathbb{N}$ that can be completed with a probability of at least p_k :

$$b_k = \lfloor F_c^{-1}(p_k) \rfloor,$$

where F_c denotes the complementary cumulative distribution function of the estimated project velocity (see Figure 3), derived from the estimated velocity distribution $f(v)$.

We define the decision variables for including story s_i in set ℓ_k and completing theme t_j by set ℓ_k as $x_{i,k}$ and $y_{j,k}$, respectively:

$$\begin{aligned} x_{i,k} &\in \{0, 1\}; i \in S, k \in L \\ y_{j,k} &\in \{0, 1\}; j \in T, k \in L \end{aligned}$$

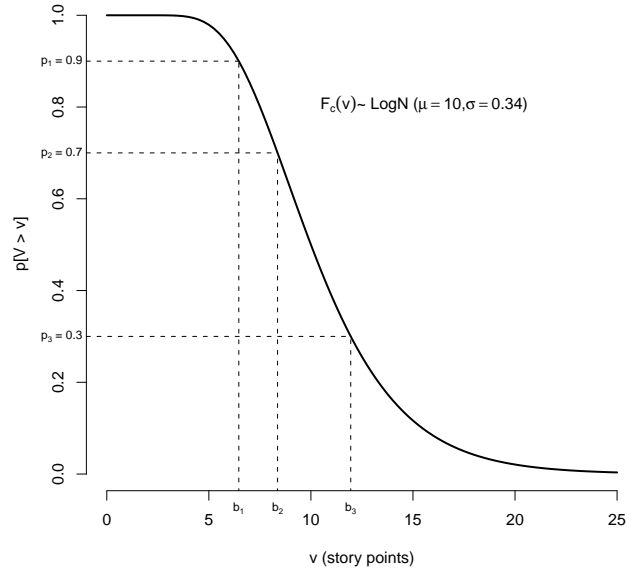


Figure 3: Completion probabilities p_i and the complementary cumulative distribution $F_c(v)$ of the release velocity v define budgets for the ‘must have’ (b_1), ‘should have’ (b_2) and ‘could have’ (b_3) sets.

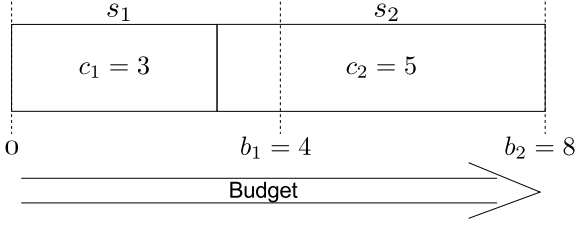


Figure 4: A two-knapsack problem with budgets b_1 and b_2 . The shown solution has a story s_1 in the ‘must have’ set and another story s_2 in the ‘should have’ set. Story s_1 is counted towards the budget of both knapsacks, and story s_2 uses the left-over capacity of the first knapsack to fit into the second.

Now, we optimize the following objective function:

$$\begin{aligned} \max \quad & \sum_{k \in L} \sum_{i \in S} x_{i,k} p_k u_i + \sum_{k \in L} \sum_{j \in T} y_{j,k} p_k u_j \\ \text{s.t.} \quad & \sum_{i \in S} \sum_{h=1}^k c_i x_{i,h} \leq b_k \quad \forall k \in L \end{aligned} \quad (1)$$

$$\sum_{k \in L} x_{i,k} \leq 1 \quad \forall i \in S \quad (2)$$

Constraint (2) ensures that the decision variable $x_{i,k}$ is set to 1 only for the set in which story s_i is included. The constraints in (1) are formulated in such a way that the story sizes included in sets prior to ℓ_k are included when evaluating the budget for knapsack κ_k . This is illustrated in Figure 4.

The dependencies between completing themes and completing the individual stories within themes are accounted for by introducing a dummy decision variable

$$z_{j,k} \in \{0, 1\}; j \in T, k \in L$$

such that $z_{j,k} = 1$ iff all stories in theme t_j are included in knapsack κ_k . To express this mathematically, we need the following constraints:

$$\left(\sum_{i \in S} \sum_{h=1}^k a_{i,j} x_{i,h} \right) - e_j z_{j,k} \geq 0 \quad ; \forall k \in L \forall j \in T \quad (3)$$

$$\left(\sum_{i \in S} \sum_{h=1}^k a_{i,j} x_{i,h} \right) - z_{j,k} \leq e_j - 1 \quad ; \forall k \in L \forall j \in T \quad (4)$$

Where $a_{i,j} = 1$ if story s_i is included in theme t_j and $a_{i,j} = 0$ otherwise, and e_j is the number of stories in theme t_j (i.e. $e_j = \sum_{i \in S} a_{i,j}$). If at least one of the stories belonging to theme j is not included in knapsack κ_k , $\sum_{i \in S} \sum_{h=1}^k a_{i,j} x_{i,h} < e_j$, in which case (3) ensures that $z_{j,k} = 0$. Similarly, if all stories belonging to theme t_j are completed in knapsack κ_k , $\sum_{i \in S} \sum_{h=1}^k a_{i,j} x_{i,h} = e_j$, in which case (4) ensures that $z_{j,k} = 1$. Finally, to make sure that $y_{j,k}$ is true iff $z_{j,k}$ is the first (in terms of k) for which $z_{j,k} = 1$, we add the following constraints:

$$y_{j,1} = z_{j,1} \quad \forall j \in T \quad (5)$$

$$y_{j,k} = z_{j,k} - z_{j,k-1} \quad \forall j \in T \forall k \in L - \{1\} \quad (6)$$

To complete the model, we note that if there are precedence relations, $i \prec j$ (i precedes j), they can be represented as

$$x_{j,k} - \sum_{h=1}^k x_{i,h} \leq 0 \quad \forall i \prec j \forall k \in L \quad (7)$$

Both technical and preference precedence relation are implemented in the same way, using constraint (7).

4. Theme valuation

The objective function of our optimization model combines theme and story values to obtain the total business value. This makes the theme values difficult to evaluate because on the one hand the value of completing a theme is an addition to completing the contained stories, whereas on the other hand all values should be on the same commensurable scale. We propose three theme elicitation approaches: constant theme value, ordinal evaluation and an indifference method.

A constant value c is appropriate if all themes have approximately the same business value to the customer. Ordinal evaluation of theme values is based on ranking the themes in an ascending order: the theme with the lowest business value is ranked at place 1 and the theme with the highest business value is ranked at place m . The theme value that is subsequently assigned to each of the themes should satisfy the ordering relation \mathcal{R} . A simple approach would be using a linear transformation function $h(\pi) = c\pi$, where π is the permutation of the theme index vector according to \mathcal{R} (i.e. the j -th element of π denotes the rank of theme j). For example, if we have three themes, and our customer ranks them as $t_3 < t_1 < t_2$ (i.e. t_3 is least important), then $\pi = (2, 3, 1)$ and hence the value of t_1 would be $2c$.

The indifference method allows direct evaluation of the theme value. The idea is to consider a theme which (hypothetically) is one story away from being completed, and ask the customer whether he would like to complete the remaining story, or rather complete a set E of other (non-related) stories outside the theme. Such questions are asked until the customer states that he is indifferent between the two for some E . The theme value is then the difference of the sum of values of stories in E and the value of the remaining story in the theme. For example, if the last story to complete has value 3, and the customer is indifferent between completing the theme and completing three other stories with values 2, 3, and 5, then the value of the theme is $(2+3+5) - 3 = 7$. The indifference method is similar to the standard technique used in utility function elicitation (cf. [30]).

5. Velocity estimation

A formal but simple velocity estimation method is extremely important as it has been shown that project management is often overly optimistic about the width of the

90% confidence interval for project duration [31]. We propose a method that is consistent with software engineering literature from both the plan-driven and agile research communities.

We measure iteration velocity with a probability distribution on a scale of $[0, \infty)$ story points. The observed format of the distribution often corresponds to the log-normal one [32]. A choice of a log-normal distribution is also consistent with the Bayesian model in [22] as well as with the method of deriving confidence intervals for velocity proposed in [23] and with NASA SEL guidelines [33]. The Bayesian network model [22] can provide good velocity estimates, but is quite complex. We propose here a simple extension of the yesterday’s weather model [9].

Let us denote by \mathbf{v} a vector of velocity observations from previous iterations. If we have a reasonable amount of observations, say at least five, we can estimate the log-normal velocity distribution through maximum likelihood with:

$$V_I \sim \log \mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$$

where $\hat{\mu}$ is the mean of the log-transformed observations $\ln(\mathbf{v})$ and $\hat{\sigma}^2$ is the sample variance $\text{sd}(\ln(\mathbf{v}))^2$. To estimate release velocity, we view a release as a collection of n_R independent iterations. Then V_R is the sum of n_R log-normal distributions, and can be estimated using the Fenton-Wilkinson 2-moment approximation [34] simplified for equal mean and variance:

$$\begin{aligned} V_R &\sim \log \mathcal{N}(\mu_R, \sigma_R^2) \\ \sigma_R^2 &\approx \ln(\exp(\hat{\sigma}^2) - 1 + n_R) - \ln n_R \\ \mu_R &\approx \hat{\mu} + \ln n_R + \frac{1}{2}(\hat{\sigma}^2 - \sigma_R^2) \end{aligned}$$

The sample variance is overly precise in the beginning of a new project where there are only a small number (i.e. < 5) of observations, so instead of estimating it through maximum likelihood, we should take into account prior beliefs. A full Bayesian approach is possible but complex. We suggest using the following weighted sum (approximating an inverse-Gamma prior with prior degrees of freedom $\nu_0 = 2$):

$$\hat{\sigma} = \frac{\sigma_0 + n \text{sd}(\ln(\mathbf{v}))}{1 + n} \quad (8)$$

where n is the number of observations and σ_0 is a prior belief of sample error that has a weight equal to one observation of true velocity. Note that (8) is applicable only if $n \geq 2$, otherwise the sample error is not defined. It should be expected that if σ_0 is reasonably large, $\hat{\sigma}$ will decrease as more observations become available.

The equation (8) requires a prior belief to be specified. In the complete absence of velocity observations, [23] proposes using $\sigma_0 = 0.29$. This falls between the levels suggested by NASA SEL guidelines for plan-driven projects that have completed the requirements analysis ($\sigma_0 = 0.34$) and preliminary design ($\sigma_0 = 0.21$) phases, respectively. We suggest to err on the safe side and take $\sigma_0 = 0.34$. Table 1 presents rules of thumb for specifying σ_0 for several

Table 1: Rules of thumb for uncertainty in velocity, from [33] (marked with *) and [23]. σ_0 for a log-normal distribution to produce the CIs at the 90% level is given to two decimal places.

Phase	Suggested CI	σ_0
Requirements Known *	$[\hat{\mu}/2.0, \hat{\mu} * 2.0]$	0.42
Requirements Analyzed *	$[\hat{\mu}/1.75, \hat{\mu} * 1.75]$	0.34
< 2 Iterations Completed	$[\hat{\mu} * 0.60, \hat{\mu} * 1.60]$	0.29
Preliminary Design *	$[\hat{\mu}/1.40, \hat{\mu} * 1.40]$	0.21
Detailed Design *	$[\hat{\mu}/1.25, \hat{\mu} * 1.25]$	0.14
2 Iterations Completed	$[\hat{\mu} * 0.8, \hat{\mu} * 1.25]$	0.14
3 Iterations Completed	$[\hat{\mu} * 0.85, \hat{\mu} * 1.15]$	0.08
> 3 Iterations Completed	$[\hat{\mu} * 0.90, \hat{\mu} * 1.10]$	0.06

levels of uncertainty, and gives confidence intervals at the 90% level conventional in software development [31]. Note that [23] suggests that the sample error should approach 0.06 as the number of iterations increases (see Table 1).

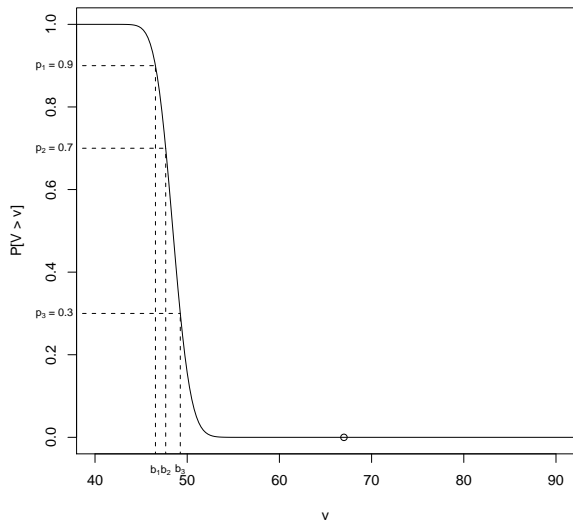
6. Example

We are involved in a research project that includes a software deliverable. In a scientific setting, developers often have other tasks that overlap with the classical separation between the developer and the customer (see e.g. [35]). Despite partly being customers for ourselves, we also have external customers that expect us to develop software artifacts that support a new way of working in the application domain of pharmacological decision making. The ‘how’ would be discovered only during the course of the project by exploring ways in which the software can support business processes.

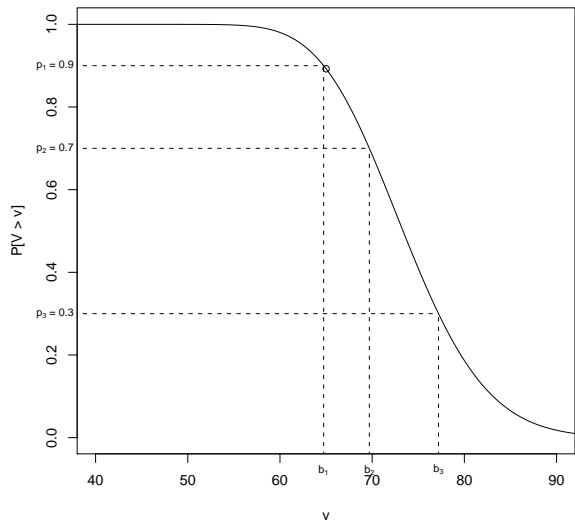
Our development environment consists of 2 teams working part-time. We programmed in Java SE 1.5 with Eclipse. All our code is available as open source. More information about the project and links to the code repository and to downloadable releases can be found at <http://www.drugis.org>. We retrospectively evaluated our velocity estimation techniques using data from this project.

We did the first release (ADDIS 0.2) as a burn-in for velocity estimation and for creating an initial end-to-end working system. Therefore we did not estimate story business values while planning the first release. During the second and third releases (ADDIS 0.4 and 0.6), we estimated story business values (scale: 1-5), story sizes (scale: 1, 2, 3, 5, 8), technical precedence relations and themes.

In the first release we had 4-week iterations, and we planned switching to 2-week iterations. With this in mind, the corrected velocity was $\mathbf{v}_1 = (8.5, 10, 9)$. With $\sigma_0 = 0.34$, the estimated velocity is $V_I \sim \log \mathcal{N}(2.2, 0.15^2)$. And, with 5 iterations, $V_R \sim \log \mathcal{N}(3.9, 0.031^2)$, the complementary cumulative distribution is shown in Figure 5(a). Velocity for the second release was $\mathbf{v}_2 = (7, 16, 17, 9, 18)$. This irregular sequence was the result of doubling the size of our development team after the first iteration. Actual velocity was well outside the predicted 95% confidence interval, showing that our decision to hire additional pro-



(a) V_R for R2 (based on \mathbf{v}_1)



(b) V_R for R3 (based on \mathbf{v}_2)

Figure 5: The complementary cumulative velocity distributions estimated for release 2 (from release 1 velocity) and release 3 (from release 2 velocity). Due to higher variability during release 2, the estimated velocity is much less certain. The \circ shows the velocity that was actually achieved.

grammers was effective. This is reflected in both increased expected velocity and increased uncertainty for the third release predicted on the basis of \mathbf{v}_2 : $V_R \sim \log \mathcal{N}(4.3, 0.097^2)$, as shown in Figure 5(b). For this release, actual velocity was slightly over b_1 , the ‘must have’ budget.

In case the predicted velocity distribution is like the one in Figure 5(a), the deterministic model we proposed previously [20] may be appropriate due to the small budget for ‘should have’ and ‘could have’. However, with greater uncertainty, such as in Figure 5(b), a probabilistic method such as the one proposed here is more suited. Note that if for some reason the actual velocity differs greatly from the predicted, replanning may be required. The planning model can then be used to suggest a revised plan for the remaining iterations.

We applied the planning model retrospectively to the ADDIS 0.6 planning problem, obtaining a plan reasonably similar to the one actually implemented. For ADDIS 0.8, we used the model to create a preliminary plan that was adopted with some modifications by our customer. Although our planning model reduced the time and effort required in release planning, managing the stories and entering them in the model was quite cumbersome due to lack of usable software that would integrate story management.

7. Computational tests

We implemented the supporting release planning model using R [36] and lp_solve [37] (using a branch-and-bound

algorithm). Our implementation is freely available online at <http://github.com/gertvv/xpplan>. The nested knapsack problem is NP-complete, but an exact solution is often feasible in practice due to small problem instance sizes. Knapsack problems are widely analyzed in the literature (see e.g. [38]), but our model has a non-standard structure. For this reason, we analyzed the running time with randomly generated problem instances with different numbers of stories and themes. The story values and sizes were sampled from $\{1, 2, 3, 4, 5\}$ and $\{1, 2, 3, 5, 8\}$, respectively. The completion probabilities were $p = (0.9, 0.7, 0.3)$, and the velocity distribution

$$V_R \sim \log \mathcal{N}(0.5c_T, 0.34^2); \quad c_T = \sum_{i \in S} c_i.$$

The themes were valued through ordinal evaluation (Section 4) with random ranks and $u_{n+j} = 5\pi(j)$. Each theme contained between 3 and 10 randomly assigned stories.

We tested running times for 10 to 50 stories (step size 5) with 2 to 10 themes (step size 2). For each problem size we ran 10 tests on an Intel Core 2 Duo 3GHz CPU with Ubuntu 9.10 and no relevant extra load during the tests. We set a time-out of 2 hours for running the model. Figure 6 illustrates the minimum, maximum, mean, and median running times of different problem sizes. The spikes in max and mean running times are due to problems that were not solved within the time limit. This occurred once for 35 stories with 10 themes, twice for 45 and 50 stories with 10 themes, and twice for 50 stories with 8 themes.

The mean running times in Figure 6 show that the problem gets harder as the amount of themes increases.

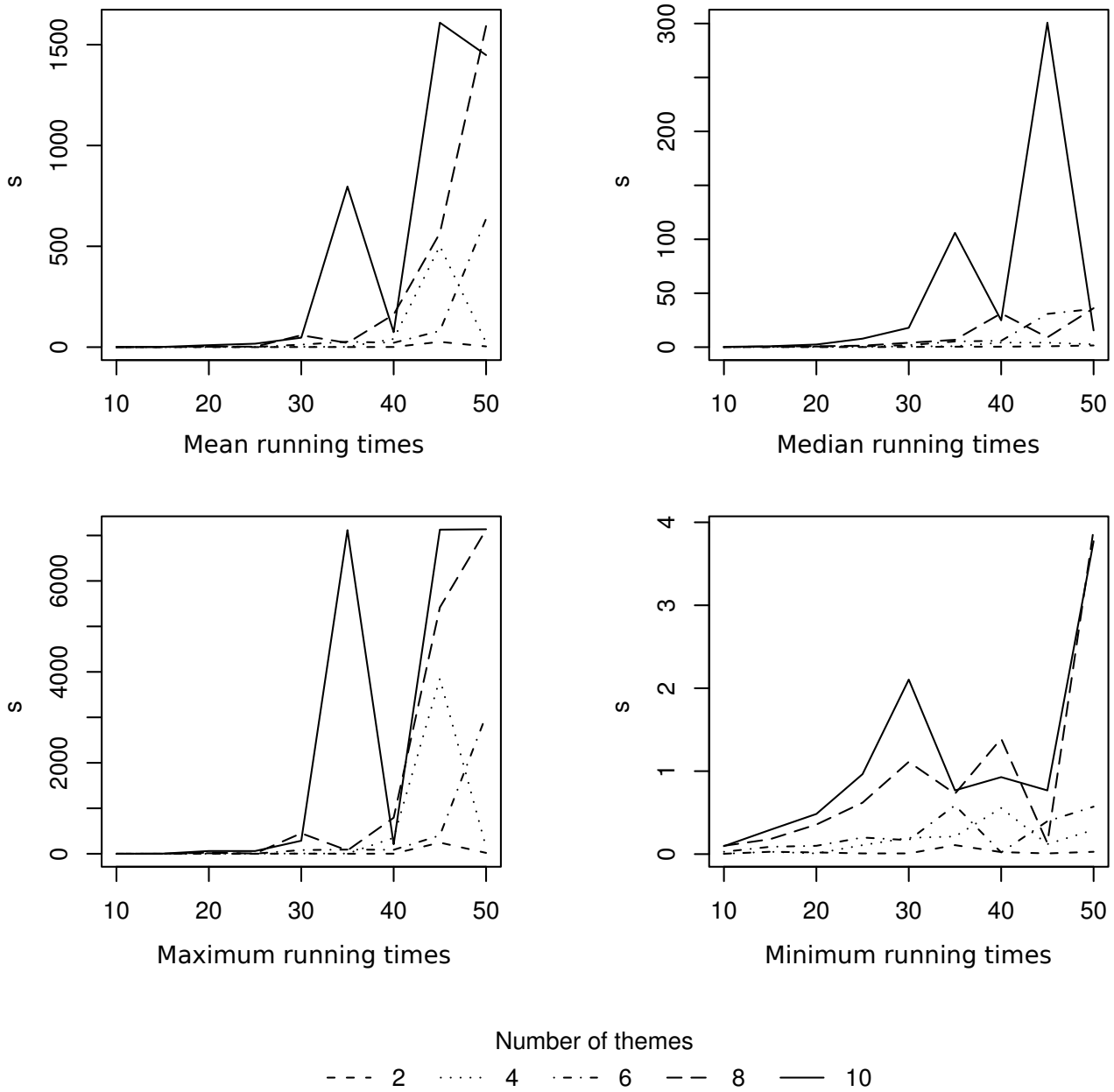


Figure 6: Running time statistics for different model sizes. Number of stories is on the x-axis.

The maximum running times indicate that problem instances with 2-6 themes were solvable within one hour. With 2 or 4 themes the median running times are very low, which hints that a lot larger problems could be solved exactly. The minimum running times are relatively low for all problem sizes, so some instances are very easy to solve even for larger amounts of themes.

8. Conclusions

Release planning in Extreme Programming (XP) can cause prioritization stress for the customer and is impractical in larger projects. To remedy this, we developed an optimization model to support release planning. Our model evaluates the stories with regard to their size, business value, and technical and preference precedence relations, and incorporates synergy effects among stories with themes. The model assumes availability of a velocity distribution. We discussed simple rules of thumb for estimating an appropriate log-normal velocity distribution and evaluated them with an example. Our experiences suggest that the model is both feasible and useful in practice.

It has been argued that there is a lack of diversity among the existing release planning models [10]. We address this by introducing a model compatible with XP values and practices [2] that is optimized for higher levels of requirements change and uncertainty in the velocity estimates by requiring less ‘up front’ specification and estimation. Most other models, in contrast, do not consider uncertainty in release velocity and operate at a more fine-grained level of detail (e.g. [12, 13, 16]). One could also consider imprecise story sizes [15], but our approach, when combined with an appropriate velocity estimation method (e.g. [22]), appears to be more general. In the framework of Boehm and Turner [19], our model can be considered more ‘agile’, while the existing models exhibit more ‘discipline’. Still, the use of our model introduces some additional discipline to enable XP outside its home ground.

The optimization model is of exponential complexity, but our computational tests showed that problems with up to 6 themes and 50 stories can be solved exactly in less than an hour with a standard PC. This is sufficient for practical use in up to medium sized projects. However, [6] suggests that for quantitative models to be useful, running time should be minimized (immediate interaction) and several good solutions may be more valuable than a single optimal one. Future research should develop and evaluate approximate methods to address this. For example, the genetic algorithm proposed in [12] solves a similar problem, and might be adaptable to our model. In addition, we chose to model dependencies through precedence relations and themes, although other relations (e.g. exclusion [13]) could be considered as well. Future research should address which model features are actually useful in supporting XP release planning, and which ones just cause

additional cognitive burden. Finally, usable software is important to enable the use of quantitative planning methodologies such as the one presented here. A simple planning model implemented in a “provotype” tool with a graphical user interface was presented and evaluated in [6], and the requirements for release planning and story management software in an agile environment were discussed in [39]. Our model can be implemented as a decision support tool in such a system. However, a planning model and software alone cannot solve the project management problems for larger projects. Other socio-technical aspects of agile software management should also be addressed. A recent review [40] has shown that currently there aren’t sufficient empirical studies on agile software management to draw firm conclusions on “goodness” of the methods. We acknowledge that this holds also with respect to our work, and empirical studies should be initiated to consider the model’s applicability on different types of development projects.

Acknowledgements

This study was partly supported by the Escher project (T6-202), which is a project of the Dutch Top Institute Pharma.

References

- [1] K. Beck, *Extreme Programming Explained*, 1st Edition, Addison-Wesley, 1999.
- [2] K. Beck, *Extreme Programming Explained*, 2nd Edition, Addison-Wesley, 2005.
- [3] T. Chow, D.-B. Cao, A survey study of critical success factors in agile software projects, *Journal of Systems and Software* 81 (6) (2008) 961–971. doi:10.1016/j.jss.2007.08.020.
- [4] P. Abrahamsson, J. Warsta, M. T. Siponen, J. Ronkainen, New directions on agile methods: a comparative analysis, in: *IEEE Proceedings of the International Conference on Software Engineering*, Portland, Oregon, USA, 2003, pp. 244–254. doi:10.1109/ICSE.2003.1201204.
- [5] A. Martin, R. Biddle, J. Noble, The XP customer role in practice: three studies, in: *Agile Development Conference (ADC2004)*, Salt Lake City, Utah, USA, 2004. doi:10.1109/ADEV.2004.23.
- [6] P. Carlshamre, Release planning in market-driven software product development: Provoking an understanding, *Requirements Engineering* 7 (3) (2002) 139–151. doi:10.1007/s007660200010.
- [7] P. S. Grisham, D. E. Perry, Customer relationships and extreme programming, in: *HSSE ’05: Proceedings of the 2005 workshop on Human and social factors of software engineering*, St. Louis, Missouri, USA, 2005, pp. 1–6. doi:10.1145/1083106.1083113.
- [8] G. Ruhe, M. Saliu, The art and science of software release planning, *IEEE Software* 22 (6) (2005) 47–53. doi:10.1109/MS.2005.164.
- [9] K. Beck, M. Fowler, *Planning Extreme Programming*, Addison-Wesley, 2001.
- [10] M. Svahnberg, T. Gorschek, R. Feldt, R. Torkar, S. Bin Saleem, M. U. Shafique, A systematic review on strategic release planning models, *Information and Software Technology* 52 (3) (2010) 237–248. doi:10.1016/j.infsof.2009.11.006.
- [11] J. Karlsson, K. Ryan, A cost-value approach for prioritizing requirements, *IEEE Computer* 14 (5) (1997) 67–74. doi:10.1109/52.605933.

- [12] D. Greer, G. Ruhe, Software release planning: an evolutionary and iterative approach, *Information and Software Technology* 46 (4) (2004) 243–253. doi:10.1016/j.infsof.2003.07.002.
- [13] M. van den Akker, S. Brinkkemper, G. Diepen, J. Versendaal, Software product release planning through optimization and what-if analysis, *Information and Software Technology* 50 (1-2) (2008) 101–111. doi:10.1016/j.infsof.2007.10.017.
- [14] A. Ngo-The, G. Ruhe, Optimized resource allocation for software release planning, *IEEE Transactions on Software Engineering* 35 (1) (2009) 109–123. doi:10.1109/TSE.2008.80.
- [15] G. Ruhe, D. Greer, Quantitative studies in software release planning under risk and resource constraints, in: *Proceedings of the 2003 International Symposium on Empirical Software Engineering (ISESE2003)*, 2003, pp. 262–270. doi:10.1109/ISESE.2003.1237987.
- [16] M. O. Salu, G. Ruhe, Bi-objective release planning for evolving software systems, in: *Proceedings of the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, ESEC-FSE '07*, 2007, pp. 105–114. doi:10.1145/1287624.1287641.
- [17] A. Ngo-The, G. Ruhe, A systematic approach for solving the wicked problem of software release planning, *Soft Computing* 12 (1) (2008) 95–108. doi:10.1007/s00500-007-0219-2.
- [18] C. Li, M. van den Akker, S. Brinkkemper, G. Diepen, An integrated approach for requirement selection and scheduling in software release planning, *Requirements Engineering* 15 (4) (2010) 375–396. doi:10.1007/s00766-010-0104-x.
- [19] B. Boehm, R. Turner, *Balancing agility and discipline: a guide to the perplexed*, Addison Wesley, 2003.
- [20] G. van Valkenhoef, T. Tervonen, E. O. de Brock, D. Postmus, Product and release planning practices for extreme programming, in: *Proceedings of the 11th International Conference on Agile Software Development (XP2010)*, Trondheim, Norway, 2010. doi:10.1007/978-3-642-13054-0_25.
- [21] A. Al-Emran, P. Kapur, D. Pfahl, G. Ruhe, Studying the impact of uncertainty in operational release planning-An integrated method and its initial evaluation, *Information and Software Technology* 52 (4) (2010) 446–461. doi:10.1016/j.infsof.2009.11.003.
- [22] P. Hearty, N. Fenton, D. Marquez, M. Neil, Predicting project velocity in XP using a learning dynamic bayesian network model, *IEEE Transactions on Software Engineering* 35 (1) (2009) 124–137. doi:10.1109/TSE.2008.76.
- [23] M. Cohn, *Agile Estimating and Planning*, Robert C. Martin Series, Prentice Hall PTR, 2005.
- [24] D. Hartmann, R. Dymond, Appropriate agile measurement: using metrics and diagnostics to deliver business value, in: *Agile Conference, 2006*, 2006, pp. 6 pp. –134. doi:10.1109/AGILE.2006.17.
- [25] J. Stapleton, *DSDM, dynamic systems development method: the method in practice*, Addison-Wesley Professional, 1997.
- [26] E. Miranda, Planning and executing time-bound projects, *Computer* 35 (2002) 73–79. doi:10.1109/2.989933.
- [27] M. Denne, J. Cleland-Huang, *Software by Numbers: Low-Risk, High-Return Development*, Prentice-Hall, 2003.
- [28] M. Denne, J. Cleland-Huang, The incremental funding method: Data-driven software development, *IEEE Software* 21 (2004) 39–47. doi:10.1109/MS.2004.1293071.
- [29] K. Dudziński, S. Walukiewicz, Exact methods for the knapsack problem and its generalizations, *European Journal of Operational Research* 28 (1) (1987) 3–21. doi:10.1016/0377-2217(87)90165-2.
- [30] R. Keeney, H. Raiffa, *Decisions with multiple objectives: preferences and value tradeoffs*, Wiley, New York, 1976.
- [31] M. Jørgensen, K. Teigen, K. Moløkken, Better sure than safe? over-confidence in judgement based software development effort prediction intervals, *Journal of Systems and Software* 70 (2004) 79–93. doi:10.1016/S0164-1212(02)00160-7.
- [32] P. Bourque, S. Oligny, A. Abran, B. Fournier, Developing project duration models in software engineering, *Journal of Computer Science and Technology* 22 (3) (2007) 348–357. doi:10.1007/s11390-007-9051-5.
- [33] NASA, *Manager’s handbook for software development*, Software Engineering Laboratory, NASA Software Engineering Laboratory, Goddard Space Flight Center, Greenbelt, MD, 1990.
- [34] L. Fenton, The sum of log-normal probability distributions in scatter transmission systems, *Institute of Radio Engineers Transactions on Communication Systems CS-8* (1) (1960) 57–67.
- [35] W. Wood, W. Kleb, Exploring XP for scientific research, *IEEE Software* 20 (3) (2003) 30–36. doi:10.1109/MS.2003.1196317.
- [36] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0 (2008). URL <http://www.R-project.org>
- [37] M. Berkelaar, K. Eikland, P. Notebaert, *lp_solve 5.5 – open source (mixed-integer) linear programming system*. URL <http://lpsolve.sf.net/>
- [38] D. Pisinger, A minimal algorithm for the 0-1 knapsack problem, *Operations Research* 45 (5) (1997) 758–767. doi:10.1287/opre.45.5.758.
- [39] J. Koponen, *Agile Release Planning in a Product Backlog Tool*, MSc thesis (2008). URL <http://www.tml.tkk.fi/~anttiyj/Koponen-Agile.pdf>
- [40] T. Dybå, T. Dingsøy, Empirical studies of agile software development: A systematic review, *Information and Software Technology* 50 (2008) 833–859. doi:10.1016/j.infsof.2008.01.006.