A Comparison of Model-based and Judgment-based Release Planning in Incremental Software Projects

Hans Christian Benestad
PREPARE Group, Simula Research Laboratory
Pb. 134, NO-1325 Lysaker, Norway
benestad@simula.no

Jo E. Hannay
PREPARE Group, Simula Research Laboratory
Pb. 134, NO-1325 Lysaker, Norway
johannay@simula.no

ABSTRACT

Numerous factors are involved when deciding when to implement which features in incremental software development. To facilitate a rational and efficient planning process, release planning models make such factors explicit and compute release plan alternatives according to optimization principles. However, experience indicates that industrial use of such models is limited. To investigate the feasibility of model and tool support, we compared input factors assumed by release planning models with factors considered by expert planners. The former factors were cataloged by systematically surveying release planning models, while the latter were elicited through repertory grid interviews in three software organizations. The findings indicate a substantial overlap between the two approaches. However, a detailed analysis reveals that models focus on only select parts of a possibly larger space of relevant planning factors. Three concrete areas of mismatch were identified: (1) continuously evolving requirements and specifications, (2) continuously changing prioritization criteria, and (3) authority-based decision processes. With these results in mind, models, tools and guidelines can be adjusted to better address real-life development processes.

Categories and Subject Descriptors

D.2.9.d [**Software Engineering**]: Management—*Productivity*; D.2.9.g [**Software Engineering**]: Management—*Software process models*

Keywords

Agile, Large Development Projects, Scrum, Practitioners' Mental Models, Repertory Grid, Case study

1. INTRODUCTION

A common best practice in software development is to deliver new functionality to the customer in an incremental fashion according to the customer's business priorities [5, 31]. One of the core management tasks in software development is therefore to make decisions about which features to prioritize for delivery in upcoming releases [3]. The activity of assigning features to subsequent releases under technical, resource, risk, and budget constraints is referred to as *release planning* [1].

ICSE 2011, May 21–28, 2011 Honolulu, Hawaii

In this paper, we compare formal release-planning with judgment-based release-planning. A formal release planning model purports to make explicit and prescribe the data and the analyses needed to generate well-founded release plan alternatives. Tools based on such models are then designed to aid in various stages of a more formal release planning process according to a given model. A systematic review by Svahnberg et al. [32] summarizes existing research on formal release planning. On the other hand, judgment-based planning (referred to as *ad hoc* planning in Svahnberg et al's review), is based on human judgment and assumes that a project is able to handle a sufficient amount of release planning concerns and tradeoffs through project members' mental processes and through more or less informal negotiations between stakeholders.

Despite the last decade's research efforts into release planning models and the availability of mature tools to support release planning, our clear impression is that most software organizations today employ pure judgment-based release planning. The situation seems analogous to the related field of software cost estimation: Despite extensive efforts in developing formalized effort estimation models, by far the most widely used estimation method today is human judgment-based estimation [10].

We assume that a critical factor for the acceptance of a release planning model is the degree of match between the concerns accounted for by expert planners, and the input prescribed by the model. If a model does not account for the most important concerns of the expert planners, a model is unlikely to be accepted. Likewise, if a model requires input data that is difficult to collect, or the input is considered irrelevant by the experts, model acceptance is unlikely. Our research question, posed in the context of large, bespoke software development projects, is:

> *To what extent do the concerns accounted for by experts in judgment-based release planning match the prescribed input to release planning models?*

There are many drivers in successful technology transfer [23], several of which go beyond a solution being technologically superior (e.g., whole product thinking [17]). Investigating the degree to which a model and tool addresses actual practices is a first step in identifying inhibitors and promoters in technology transfer [21]. Further, it is important to identify the target practitioners for the technology and to balance a supportive versus a normative agenda: A mismatch between tool and practice could indicate a need to improve release planning models so as to better accommodate planning situations, but could also be attributed to deficiencies in judgment-based planning practices. With the research question answered, we expect to understand better the causes for the perceived sparse usage of release planning models, and to be able to point to specific areas of improvement for existing practices and tools that purport to support release planning.

2. RELATED WORK

Our research question has been touched upon in studies that typically evaluate a given new model against some benchmark judgment-based process. The following papers covered in Svahnberg et al.'s systematic review on release planning models [32] are relevant in this respect.

Karlsson and Ryan [11] conducted a study at Ericsson Radio Systems. They reported positive user feedback for a model based on the Analytic Hierarchy Process [29] that systematically compares pairs of features with respect to cost and value. Two areas for model improvement were identified: The pair-wise comparisons were perceived as laborious by the involved stakeholders, and interdependencies between requirements were not accounted for.

Amandeep et al. [2] reported from a study at Corel Corporation. In judgment-based planning, requirements were insufficiently described, requirement interdependencies were not clarified, and effort estimates arrived late and were inaccurate. The authors reported that introducing systematic release planning remedied these alleged process deficiencies.

Ruhe and Momoh [27] reported from a study at Trema Group that judgment-based release planning was inefficient due to continuously changing requirements and the costs of stakeholder negotiations. After introducing model-based release planning, the planning process required significantly less effort and stakeholders were more satisfied. These positive effects were attributed to tool support for handling changing requirements and for facilitating the involvement of all stakeholders in the decision process.

The model proposed by Regnell et al. [22] requires that quality indicators, benefit breakpoints, and cost barriers can be identified for particular product domains. In a case study at a vendor of mobile handsets, the feasibility of collecting these parameters was verified for six sub-domains. A drawback with the model was that it did not directly support selection and tradeoffs between multiple quality indicators.

Heikkilä et al. [8] proposed a five step process on top of an existing release planning model and tool ([24]), and conducted a survey to collect feedback from a case study of this process. One of the survey questions "This method of prioritization allows me to express my needs" resembles our research question, and received a score of 4.5 on a 6-point Likert scale. Three of the stakeholders expressed that they "had difficulties differentiating between the criteria".

In summary, while some of these studies indicate a reasonable match between the concerns accounted for by experts and the input prescribed by release planning models, there also seems to be room for substantial improvement. Apart from Heikkilä et al.'s survey, we are not aware of attempts to systematically collect empirical evidence on this topic.

Our study had three stages. First, we investigated factors considered in release planning practices. Second, we catalogued the input factors that are prescribed by release planning models. Third, we compared factors obtained in the first stage with those obtained in the second stage.

3. JUDGMENT-BASED RELEASE PLANNING

In this section, we describe the methodology used and the results of the first stage of our study.

3.1 Study design

To describe concerns considered by experts in judgment-based release planning, we investigated release planning practices in three projects in three different organizations in the Norwegian public sector. One of the projects is, at the time of writing, the largest software development project in the country in this sector. We chose this project, Project A, as the primary representative for judgment-based release planning. The two other projects may be described as variations and extensions to the reference project.

Project A is characterized by extensive parallel development due to time pressure, complex business rules, and fixed price elements in vendor contracts [7]. The functional scope is defined by about 300 user stories [4], acting as short descriptions and placeholders for development tasks requiring effort in the magnitude of one person-year. The project selects, designs, builds, and deploys a subset of the user stories every four months, giving three releases per year. Ten Scrum teams [30] consisting of eight developers on average, deliver production-ready code in three-week iterations between these release points. A six-week planning period precedes each release period, wherein user stories are elaborated upon, possible interdependencies between development tasks are clarified, priorities are set, development effort is estimated, and release plans are created.

The project subcontracts three different vendors to staff the Scrum teams. Each vendor works with a designated Product Owner, which is the business side's representative for the relevant system areas. We interviewed the three Product Owners, the development manager and the lead architect. These people were perceived as possessing central roles in the planning process.

Projects B and C have similar characteristics, both being bespoke development projects in the public sector and using Scrum practices and periodic releases. However, these projects are smaller, around twenty participants, and are in a phase of maintenance and evolution. The development project managers of these projects were interviewed.

3.2 Repertory grid interviews

The interviews were conducted according to the repertory grid interviewing technique [6, 9]. This technique is well suited when the goal is to obtain insights in people's current understanding and beliefs. The technique is based on Kelly's personal construct theory [12] which implies that people try to understand concepts and relationships in the world in terms of bipolar constructs. It is an open interview technique, which minimizes interviewer bias and enables efficient analysis using a combination of qualitative and quantitative methods. In a repertory grid session, the interviewer presents the interviewee with a topic. The topic presented orally to the interviewees in our case was

> *Things you take into consideration when assigning priorities to development tasks*

The interviewee is then asked to give concrete, self-experienced examples to paint a picture of his or hers view of the topic. Each example, dubbed an element in the repertory

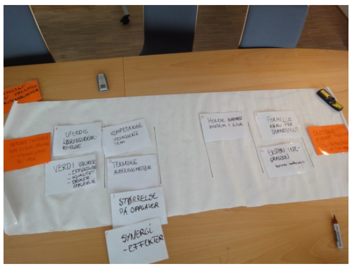

Figure 1: Rating mat with elements (white) and bipolar construct (colored).

grid technique, was written down on a cue card. It is recommended that the granularity of the examples should be such that 7–12 elements can be retrieved from the interviewee. This number was suggested to the interviewees.

When the retrieval of elements has come to an end, the repertory grid technique prescribes a session where three random elements are selected at a time. For each three elements, the interviewee is asked to give a characteristic of two elements that contrasts them to a characteristic of the third. For example, if the three elements "interdependencies", "user would suffer if feature is left out", and "target architecture" were to be compared, the interviewee might characterize the first and the third elements as "technical", while characterizing the second element as "functional". The resulting contrast pair "Technical–Functional" is said to constitute a bipolar construct in the interviewee's mental model. The technique assumes that it is possible to rank all elements on a scale defined by such a bipolar construct. Hence, to clarify how the interviewee thinks about each element, he is asked to place the elements relatively to each other according to how much he perceives they relate to one or the other contrast; see Fig. 1. The procedure of selecting three random elements is repeated until no further constructs can be identified.

It is often sensible to ask the interviewee to rate the elements according to a number of researcher supplied constructs of special interest to the researchers [6, 9]. At the end of each session, we asked the interviewees to rate each element according to the following supplied constructs:

- Easy to assess – Difficult to assess
- Can be assessed early – Must be (re)assessed late
- Not so important factor – Important factor

These constructs were provided to translate the interviewees' perceptions into immediate improvement initiatives: It makes sense to start with elements that are classified as *Important factor*, *Can be assessed early*, and *Easy to assess*.

3.3 Aggregation of grids

When the construct system of more than one person is to be analyzed, individual grids must be aggregated. A special form of content analysis is recommended for this purpose [9, 13]. To aggregate the elicited elements, each of the authors grouped the elements into inductively defined categories, and assigned a descriptive name to each category [9]. We then compared the two resulting categorization schemas and identified any categories that had similar meaning between the two authors. The proportion of elements categorized equally with regards to these categories to the total number of elements belonging to matching categories was tracked as reliability measure RE1. The proportion of matching elements to the total number of elements was tracked as RE2. RE2 will always be less or equal to RE1, because RE1 only regards the elements that are included in the matching categories, while RE2 includes the total number of elements.

Table 1: Agreement scores

Element categories		Construct categories	
RE1	92%	RC1	77%
RE2	88%	RC2	75%
RE3	86%	RC3	80%

To produce a final categorization schema, the authors negotiated the meaning of each remaining category. Disjoint categories were either merged into the already agreed categories (and the semantics adjusted accordingly), or new categories were agreed upon. The elements were then placed accordingly.

To measure the reliability of the finalized categorization schema, each of the authors repeated the categorization of elements into the agreed categories, after three months of elapsed time, and the proportional agreement (number of matching assignments to total number of elements) was tracked as RE3.

To aggregate the elicited constructs, the procedure was repeated, this time using the constructs as the basis for categorization. The corresponding reliability was measured and denoted RC1, RC2 and RC3.

Table 1 gives the agreement measures defined above. The numbers indicate reasonably well-defined and disjoint categories.

3.4 Results: Categories

A total of 42 elements were elicited in Project A, and content analyzed into the following seven categories:

- Design specification's clarity and linkage to overall system goals
- Benefit for user
- Dependencies within solution
- Resource availability and project roles
- Costs
- External requirements
- Technical and functional sustainability

A total of 22 constructs were elicited in Project A, and content analyzed into the following five construct categories:

- Self-imposed vs. externally imposed concerns
- Engineering vs. soft concerns
- Functional vs. technical concerns
- Execution vs. goal-oriented concerns
- Agile vs. plan-based concerns

The construct categories constitute a five-dimensional space of concerns accounted for by expert planners in Project A, while the element categories summarize the concerns they focused on in the project.

Project B and C resulted in 8 elements and 5 constructs in each of the two projects. These were not included when generating the categories above, but are discussed separately in cases where they represent variations from the categories.

Hard constraints	Technical constraint	Requirement dependencies
		Quality constrains
	Budget and Cost constraints	
	Resource constraints	
	Effort constraints	
	Time constraints	
Soft factors	Stakeholders influence factors	
	Value factors	
	Risk factors	
	Resource consumption factors	

Table 2: Taxonomy[18]

4. FACTORS PRESCRIBED BY MODELS

The second stage of our study was to catalog input factors that are prescribed by release planning models. We compiled such factors from the primary papers included in a recently published systematic literature review on release planning models by Svahnberg et al. [32]. The factors were then organized according to a taxonomy developed as part of Svahnberg et al.'s review, see Table 2. Section 5 discusses these concrete factors in more details.

We selected the most comprehensive release planning model in the review as a reference model, to which other models may be described as variations and extensions. The specific reference model chosen was the model underpinning the most recent (2010) version of the tool ReleasePlanner [24]. This model most comprehensively covers the categories of Svahnberg et al's taxonomy. It represents the EVOLVE-family of release planning models, treated in 16 out of 24 primary papers in [32].

5. COMPARISON

The third stage of our study compared the factors prescribed by models (Stage 2) and the factors elicited from actual projects (Stage 1). The following discussion is organized from the point of view of the categories generated from the elicited elements of Stage 1. This gives the empirically elicited factors precedence over those derived from models. This is a deliberate position in our current study.

In the following, we will discuss each category in turn. For each category, we plot the elements according to their ranking on our supplied Easy to assess – Difficult to assess and Important – Less Important constructs (Figures 2 through 8). It is reasonable to assume that important planning factors that are also difficult to assess constitute areas that deserve particular focus in release planning practices. For clarity of presentation, we omit the third supplied construct from this discussion.

5.1 Design specification's clarity and linkage to overall system goals

Items to be scheduled for future releases go under several names: features, user stories, use cases, tasks etc., and vary between different development methodologies and projects. In any event, an item to be scheduled represents a cohesive unit of work that improves some functional or technical quality of the system.

In Project A, items to be scheduled are briefly described user stories requiring an average development effort of one person-year. During release planning, user stories are elaborated upon and some are divided into a few smaller stories. In the project, concerns were raised as to the degree of elaboration and specificity of the user stories. These concerns are what ended up in the category *Design specification's clarity and linkage to overall system goals*. All concerns belonging to this category were thought to be important; see Fig. 2. If user stories were not specified and designed at a sufficient level of detail, user benefits and costs were difficult to assess, and appropriate release scheduling became difficult. To achieve sufficient throughput, the scheduling activity had to be performed in parallel with the specification process. The expert planners therefore had to accept a degree of uncertainty associated with the understanding of user stories to be scheduled, and had to update the plan as new information and understanding emerged.

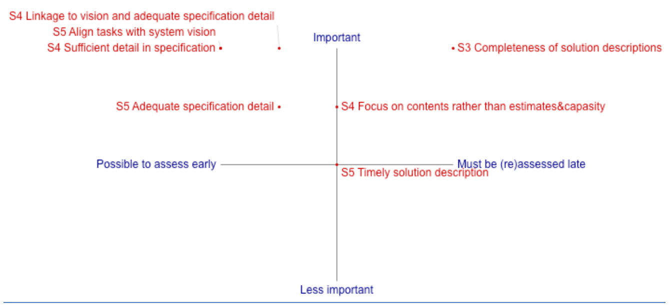

Figure 2: Category: Design specification's clarity and linkage to overall system goals

In the reference model, planning items may be named freely to cater for different methodologies. They are assumed to be described at a level of precision that enables the stakeholder to evaluate facets of added value implied by the feature. It is also assumed that realistic resource estimates can be given and that dependencies with other tasks can be identified.

These observations suggest the following possible mismatch:

- 1. Release planning models assume a shared understanding of features and tasks between stakeholders, while in reality, such understanding may evolve through a continuous process of knowledge generation and sharing.

For Project B and C, no issues with task descriptions were mentioned. It is reasonable to believe that this difference is due to these systems being in a phase of maintenance and evolution, where development tasks could be described relative to an existing, tangible system.

5.2 Benefit for user

The extent to which user stories contributed to core functionality and added value were prime concerns with the interviewees in Project A. In our content analysis, such concerns ended up in the category *Benefit for user*. The high-level user stories had been assigned a preliminary priority during the project inception phase. This initial priority was guiding, but not binding during the actual release planning.

Being familiar with the overall release goals was essential to the stakeholders when they assessed benefits and priorities. The overall goal of a release ranges from business-level requirements to architectural restructuring, and a user story was deemed beneficial to the extent that it contributes to the fulfillment of these goals. Once the link to release goals was established, concerns of users' productivity were considered. Fig. 3 summarizes the concerns mentioned by the subjects.

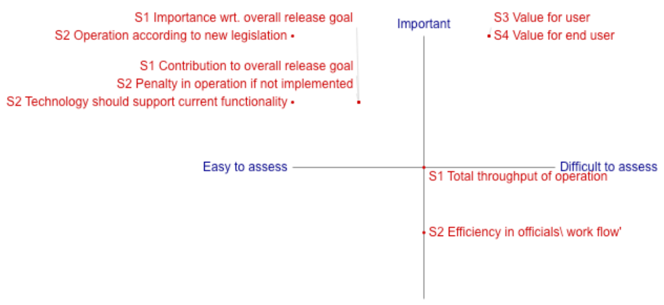
S1 Importance wrt. overall release goal
S2 Operation according to new legislation
Important
S3 Value for user
S4 Value for end user
S1 Contribution to overall release goal
S2 Penalty in operation if not implemented
S2 Technology should support current functionality
Easy to assess
Difficult to assess
S1 Total throughput of operation
S2 Efficiency in officials\ work flow'

Figure 3: Category: Benefit for user

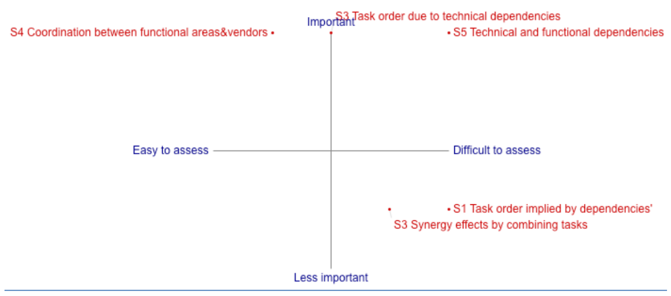
S4 Coordination between functional areas&vendors
Important
S3 Task order due to technical dependencies
S5 Technical and functional dependencies
Easy to assess
Difficult to assess
S1 Task order implied by dependencies'
S3 Synergy effects by combining tasks
Less important

Figure 4: Category: Dependencies within solution

For project B and C, which were both in a phase of maintenance and evolution, reducing cost of IT operation was an additional, important aspect of benefice.

To prioritize user stories, the product owner in Project A uses an ordinal assignment scheme [3], with three levels: A user story is *critical* if a release goal cannot be reached without it (but initial design may be simplified to achieve acceptable cost), *desired* if the users story implies that the users can increase their productivity, and *optional* if the user story is useful, and not directly characterized by the former two criteria.

In the reference model, value and benefits are encompassed by the notion of *value criteria*. Value criteria are at the core of assessing or maximizing the value of a release. Indeed, the goal of release planning models is to schedule features so that value is optimized under given constraints. The reference model allows for any number of value criteria to be defined by the model user. Stakeholders are invited to use a 9-point scale to rate each feature on each value criterion. Each criterion can be weighted to allow the scheduling algorithm to put more or less emphasis on certain criteria. In [28], the value criteria are handled as individual optimization criteria, and the trade-off between the individual optima is left to the expert planner. Other methods than 9-point scales have been used; e.g., the analytic hierarchical process (AHP) and pair-wise comparisons to come up with a total ranking of the features [11].

Other variants over the reference model include the model in [22] which differentiates between useless, useful, competitive, and excessive benefit. Stakeholders can be asked about their judgment on when the benefit reaches a market breakthrough. Financial measures, such as *net present value* [16] and *revenue* [33] have been proposed as value and optimization criteria. Note that all these methods, including that of the reference model, assume non-trivial human judgments as input.

These observations suggest the following possible mismatches:

- 2. Release planning models assume stable criteria over all releases for value assessment, while in reality, the value of a feature may be tied to goals stated for each release.

- 3. Release planning models assume that the inclusion of a feature is negotiable, while in reality, critical features may always have to be included, and then possibly with an on-the-fly modification in the design to reduce cost.

The latter point could partly be accounted for by the reference model through pre-assignment of features to releases. However, the possibility for dynamically reducing feature cost is not integrated into the scheduling algorithm.

5.3 Dependencies within solution

The interviewees mentioned that various types of dependencies between the user stories influence priorities. Such concerns were placed in the category *Dependencies within solution*. Fig. 4 summarizes the concerns mentioned for this category. Clarifying dependencies was intertwined with the design process (Section 5.1), and was perceived particularly challenging because interdependencies normally involved several competence areas within the system.

In release planning models, requirement dependencies refer to constraints on the order with which features can or should be developed. In the reference model, a coupling dependency refers to a situation where two features must be developed in the same release, while a precedence dependency entails that the development of one feature must precede the other. In [15] the authors use a "not later than"-dependency, which is equivalent to stating a disjunction between a coupling dependency and a precedence dependency. In [33] one adds an exclusion dependency, signifying that two features cannot be developed in the same release. These authors also allow for explicitly modeling synergy effects and added costs implied by two features being developed together.

More direct variants to the reference model include [20] where fuzzy logic is used to allow stakeholders to express a degree of dependency; for example, to express the strength of a synergy effect if two features were to be developed together, or the degree of a stakeholder's belief that a dependency actually exists. In [33], synergy effects between pairs of features are modeled explicitly.

These observations suggest the following possible mismatch:

- 4. Release planning models assume clearly defined, pairwise interdependencies, while clarifying interdependencies in reality may involve substantial creation and interchange of system knowledge, the need for which is only apparent during the more detailed elaboration phases.

5.4 Resource availability and project roles

In Project A, it is apparent that people possessing deep domain knowledge and system knowledge quickly become critical resources. It was also perceived that there are not enough people with this competence to serve the demands of the other actors in the project. These concerns were grouped into the category *Resource availability and project roles.*

The reference model requires estimates of resources needed to develop new features. Several resource types can be specified and each of them can be estimated separately.

Variants relative to the reference model include [26], where

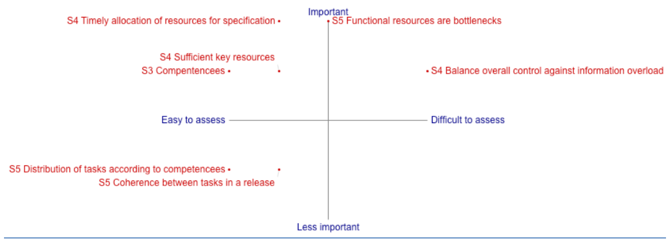

Figure 5: Category: Resource availability and project roles

effort estimates are generated by an integrated simulation model, taking parameters such as product size and complexity as input. In [19], one allows for uncertainty in the resource estimates, by letting the user define a triangular probability function for the resource estimates. Again, [11] uses AHP and pair-wise comparisons to assess the resource requirement for each feature. In [22] one elicits stakeholders' cost barriers, i.e., the level of quality at which costs start to rise quickly with further improvements to the quality level.

Models can account for different resource types, but they focus on development capacity. Interestingly, although it was perceived in Project A that a lack of personnel with domain and system knowledge presented serious bottlenecks in the project, needed capacity and resources were, in fact, estimated with respect to development effort, only. It would seem that both models and practices should broaden their scope beyond capacity for development. These observations suggest the following tentative mismatch:

- 5. Models focus on development capacity as the key resource constraint, while in reality, combined domain and system knowledge is a critical resource throughout planning and development. Effort required by various resource types may be perceived as being difficult and time consuming to estimate, especially at the granularity of individual features.

5.5 Costs

In Project A, the release dates were fixed and absolute, and development capacity was estimated for each of the three development vendors. Concerns around the constraints this incurred in terms of e.g., fitting budgets, time, and contracts were categorized in the category *Costs*. Fixed deployment dates and estimates of development capacity were major planning constraint in Project B and C as well.

In the reference model, the model user can define any number of resource types, and assigns the capacity for each resource type in each release. The scheduling algorithm aligns these constraints with estimates of resources needed per feature. In the reference model, a feature can be pre-allocated to a certain release, hence accounting for specific timing requirements. The projects empirically investigated in this study are well catered for in this respect. For example, by defining the capacity of each vendor as a resource type, the cost structure for Project A's releases may easily be expressed in the reference model.

Variants of the reference model include [33], which allows for a capacity penalty to be incurred if resources are moved between resource types (team transfer). In [19] one allows

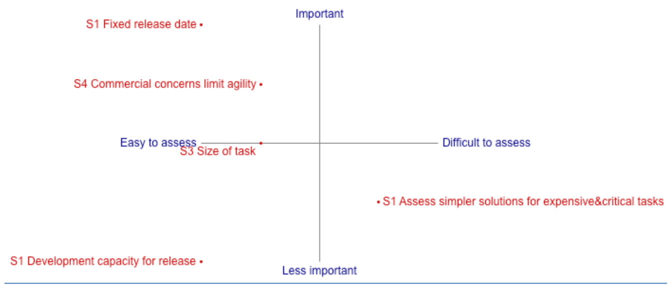

Figure 6: Category: Costs

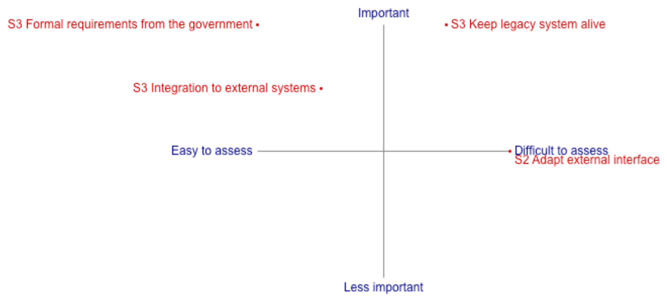

Figure 7: Category: External requirements

for uncertainty in the assessment of resource capacity by letting the user define three-point estimates of capacity. In [26], the authors allow for a constraint on duration of each release. This constraint should then be balanced against constraints on effort and quality.

5.6 External requirements

In Project A, there is one product owner for each of the three vendors, and each product owner is made fully responsible for priorities and scheduling of tasks across a vendor's teams. The interviewees made a clear distinction between external and internal stakeholders. Most requirements from external stakeholders were perceived as mandatory. Likewise, in Project B, requirements from a certain external stakeholder always received highest priority. These concerns of requirements that come from outside the project and which often conflict with the flow of the project were rather pronounced and got classified in the category *External requirements*.

In the reference model, value estimates from certain stakeholders may be given more weight so that the scheduling algorithm emphasizes the opinions of more important stakeholders. In addition, different stakeholders can be invited to assess different sets of features, and different stakeholders can be invited to assess different aspects of value. Thus models support rather elaborate patterns of how stakeholders influence release planning.

These observations suggest the following possible mismatch:

- 6. Models assume that several stakeholders may have an explicit impact (through voting) on the release plan, while in actuality, planning decisions may follow a hierarchical pattern.

5.7 Technical and functional sustainability

In Project A, development tasks describing improvements

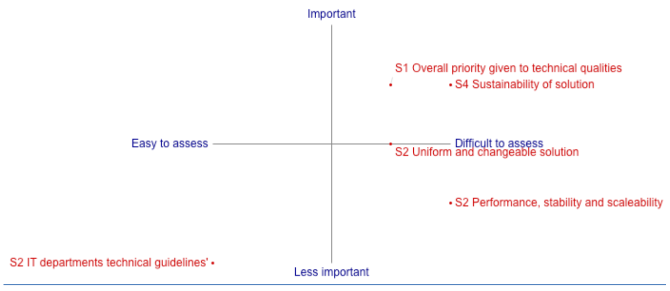

Figure 8: Category: Technical and functional sustainability

to technical qualities are defined as separate planning items entering the same prioritization procedure as do functional requirements. This approach is consistent with the reference model; see below. However, at the time of study, the development groups were under pressure to deliver the required functionality for the next release. The project explicitly expressed a policy where technical qualities had to be deemphasized in this period, but made it clear that such qualities should receive more attention in later releases. Hence, constraints on quality were stated qualitatively, as part of clarifying the release goals. In Projects B and C, the situation was similar. For some releases, the improvement of technical qualities was stated as an important release goal, and development tasks to increase such qualities received high priority. Such concerns were categorized under the category *Technical and functional sustainability.*

In general, quality constraints are requirements on qualities such as dependability, usability, performance, and maintainability. The reference model does not explicitly allow for quality constraints. However, the improvement of a given technical quality can be treated as any other feature, and traded off against other features.

Variants to the reference model include [26], where quality is measured as the number of defects per KLines of code, and the model allows for an upper limit on this measure for a specific release. This quality constraint must then be balanced against constraints on effort and duration. A simulation model is employed to estimate the quality for a given release plan alternative.

These observations suggest the following possible mismatch:

- 7. Release planning models suggest that the quality of a new version can be predicted, and checked against quantitative quality constraints. In reality, a project might not have the required resources and measurement infrastructure to do so, and may instead employ qualitative descriptions of quality goals.

5.8 Risk factors

Risk factors assess the likelihood of undesired events and their consequences in release plan alternatives. In the reference model, risk can be dealt with by quantifying the negative value of an undesired event inherent in a feature and defining a total amount of acceptable risk per release. The method in [15] lets the stakeholder assess the risk associated with each release plan alternative, using a qualitative risk assessment framework.

In Project A, none of the interviewees explicated risk as a concern during release planning. However, from informal discussions we were made aware that the project had clear procedures for dealing with threats that could lead to undesired events and therefore present risks. Nine different types of threats were identified in the project. If a particular risk was identified, it was assigned to one of these nine pre-defined threat types, and a most likely cost estimate for the risk was obtained. This method of quantifying risk closely matches the reference model: The types of threats may be identified as resource types, and a maximum acceptable risk per release may be assigned. In Projects B and C risk factors were not mentioned by the interviewees.

5.9 Analysis of constructs

We now turn to the 9 construct categories. As for individual constructs, construct categories are interpreted as ordinal scales, on which elements can be ranked. For our comparison of planning models with actual planning, we discuss the extent to which model-based release planning reflect release-planning concerns along these scales.

Self-imposed vs. externally imposed concerns. Some concerns originated from the project itself, while others had been imposed on the project by external parties. Externally imposed concerns usually took priority over self-imposed concerns. In the reference model, priority differences can be accounted for by assigning different priorities to different stakeholders, and by pre-allocating certain tasks to specific releases. However, it is not obvious how to appropriately take account for the instability and unpredictability of these externally-imposed concerns.

Engineering vs. soft concerns. Some concerns mainly related to the efficient and rational engineering of the software system, while some related to the psychosocial and collaborative working conditions in the project. The model taxonomy (Table 2) distinguishes between soft and hard constraints; however the soft factors in this dichotomy are still engineering type factors in terms of our construct category.

Functional vs. technical concerns. Expert planners distinguished between priorities for functional qualities and priorities for technical qualities. Trading off between the two was a continuous challenge. Although technical qualities can be accounted for by release planning models, as described in Section 5.7, the models provide little support in making actual trade-offs.

Execution vs. goal-related concerns. Some concerns were related to the alignment of tasks with the system vision and release goals, while others were concerned with preparing for efficient development. This construct category is similar to the distinction between strategic vs. operational release planning, where, according to the definition provided in [32], the former is input to a series of the latter. However, an observation from our study is that goal-oriented and execution-oriented concerns were not easily separated.

Agile vs. plan-based concerns. Some concerns originated mainly from agile principles while some originated mainly from plan-driven principles. For example, a key concern related to agile-based principles was to timely describe features and designs to a sufficient level of detail for prioritization and effort estimation. As described in Section 5.1, the concern of evolving feature descriptions and design specifications are not well accounted for by release planning models.

6. DISCUSSION

By its focus on contrasts, the repertory grid technique aids in putting issues into a larger perspective. We found that release-planning models focus their strengths within limited areas of the space defined by the construct categories that we elicited: Models tend to focus on internally rather than externally imposed concerns, on engineering-type rather than soft concerns, and on plan-based rather than agile-based concerns. Accounting for both functional and technical requirements in the planning process is not well supported, and neither is accommodating for both strategic and operational concerns. Some of these limitations constitute possible improvement areas for release planning models and associate tools, while others might be beyond the scope of such tools. In the latter case, we believe it is important that model and tool users are aware of concerns that are not covered to be able to compensate when determining the eventual release plans.

From the analysis of element categories, we summarize the following to be areas that both expert-based and model-based planning consider: costs, resource capacities, interdependencies between features, risk, and multiple aspects of value. The fact that factors coincide at this level of abstraction indicates that it can be useful to study detailed planning factors for one approach in view of the other. When detailing such factors, our study pinpoints specific issues that require attention:

Evolving requirements and design. While model-based release planning presumes crisp definitions of candidate features, the planning phase in Project *A* included substantial effort to understand the features, their design specifications and their interdependencies. One viewpoint might be that such activities should precede the release planning activity. However, there are reasons why release planning must sometimes occur in parallel with analysis and design activities. First, projects may, pragmatically, allow designs of "mandatory features" to be simplified if capacity is about to be reached. The need for such redesign activities will only surface during the planning activity. Second, to achieve the required throughput of design specifications and release plans, letting the two activities occur in parallel may be the only option. An insight from this discussion is that continuous replanning may be necessary both with judgment-based and model based release-planning; cf. [25].

Ruhe and Momoh [27] reported from an organization experiencing late and unclear requirements, where the introduction of model-based release planning led to improvements in the requirements process. That such favorable biproducts occur is probably situation dependent, and probably also depends on the organization's willingness to change requirement analysis practices. Even with positive side effects like those reported by Ruhe and Momoh, we believe that being able to handle evolving requirements and designs is likely to constitute a challenge when introducing more structured release planning in organizations that use incremental development methods.

Authority-based decision processes. Planning models take into account priorities from several stakeholders, and attempt to balance satisfaction among the stakeholders. In Project *A*, the Product Owner of a given system area collects input from stakeholders, but is given the authority and responsibility for setting priorities. Possible advantages with an authority-based approach are that a steady path towards an ultimate vision for the system can be maintained more easily, and that responsibilities are more clearly defined. With a more democratic process, stakeholder satisfaction might be higher overall, due to more direct influence on the planning outcome. Although model-based release planning can be tailored to an authority-based decision process through stakeholder weights, we believe that organizations should be prepared to explicate, and challenge, their current authority systems when model-based release planning is introduced.

Priorities in the face of continuously changing criteria. Release planning models presume that criteria such as *importance* and *urgency* are properties inherent to a given feature. Hence, values assigned to these properties are assumed to remain relatively stable over several releases. In Project *A*, the goals stated for the upcoming release were heavily taken into account when setting the urgency for a given feature. For example, a refactoring task might be considered not urgent (low priority) for a release because technical qualities were down-prioritized in this release. The situation could be reversed in the next release, giving high priority to refactoring tasks.

From this, it seems that a new concept of *release focus* in release planning models would help in achieving a better conceptual match with release planning in practice. Project management and product owners could decide upon foci for the upcoming releases, while other stakeholders could assign release-independent assessments of system-specific aspects of value. With different settings for release focus, the proposed plans would also differ.

A clearer conceptual separation between value inherent to features, and focus decided for each release would probably be beneficial to resolve the ambiguity in the concepts of prioritization and priority. For example, priority is sometimes considered as the value inherent to a feature, while other times it might express the decided sequence of features in the development plan. We observed this in Project *A*, and the ambiguity is also described in [14]. Resolving such ambiguities is important in order to achieve an efficient and rational release planning process.

6.1 Future work

This study has given some insight into how experts think about costs, benefits, and resulting prioritizations. An area for future research is to understand more about how expert planners employ a multitude of criteria, coupled with uncertain information, to come up with "value" or "benefit" assessments for a given task. With a better understanding of this essential step of release planning, existing models and tools could be further improved.

6.2 Threats

The reliability in the researchers' interpretation of interview data is an obvious threat to validity in this study. In spite of audio recording, double, independent analysis and member checking, it is clear that the inductive process of generating classifications could be colored by preexisting knowledge and opinions within the researchers own belief systems. However, it is reasonable to expect that this bias would have a conservative effect, leading to the discovery of fewer discrepancies between actual planning and planning according to "the book". Hence, it is reasonable to treat the discovered discrepancies as substantive.

The validity of the study beyond the immediate study context is problematic for all empirical studies within software engineering, and for case studies in particular, due to the lack of validated constructs and theory. In this study, we attempted to shed some light on the generalizability by generating the classification in one study, and checking its robustness in two additional projects. However, the three projects were similar in many respects, but varied a great deal in size. We believe the size factor, in terms of number of people and organizational units involved in the prioritization process, is an important discriminator with respect to how release planning is and should be performed. Additional situational factors that might have influenced the results are development type (bespoke), delivery rate (frequent deliveries) and criticality (business critical). Arguments of generalizability findings are therefore suggestive only.

7. CONCLUSION

We have compared the concerns accounted for by experts in judgment-based release planning with prescribed input to formal release planning models.

We found that the two approaches cover common ground when it comes to inputs to the planning activity. Concerns related to costs, resource capacities, interdependencies, risk and value were important in both approaches. By analyzing these areas in more detail, we identified three major sources of mismatch between the two approaches to release planning

- Evolving requirements and design
- Authority-based decision processes
- Priorities in the face of continuously changing criteria

Our study indicates that developing a shared understanding of features and their possible interdependencies requires the most effort in the planning process, and that the allocation of features to releases constitute a less complex task. Nevertheless, this allocation process can require substantial resources, and it is probable that the introduction of release planning tools, in their current state, would make the planning process more efficient. However, we believe that it is possible to harmonize release planning models better with the mental models of expert practitioners, and that this closer match will increase the usability positive impact of release planning tools.

Our hope is that findings from this study can be taken into consideration by practitioners, tool developers, and researchers who are working on new or improved methods for release planning. The three areas of concern mentioned above constitute a starting point for such efforts.

Acknowledgments

The authors are grateful to the project managers and interviewees of Projects A, B, and C for giving us their valuable time and for their enthusiasm and openness during the interviews. The authors are also grateful to Günther Ruhe who provided us with access to the ReleasePlanner tool and to Dietmar Pfahl who explained the fundamentals of the tool to us.

References

[1] A. Al-Emran and D. Pfahl. Operational planning, re-planning and risk analysis for software releases. In J. Münch and P. Abrahamsson, editors, *Proc. Int'l Conference on Product-focused Process Improvement (PROFES 2007)*, pages 315–329. Springer, 2007.

[2] G. R. Amandeep and M. Stanford. Intelligent support for software release planning. In *5th Int'l Conf Product-Focused Software Process Improvement (PROFES 2004)*, page 248. Springer, 2004. Product focused software process improvement: 5th international conference, PROFES 2004, Kansai Science City, Japan, April 5-8, 2004; proceedings.

[3] P. Berander and A. Andrews. Requirements prioritization. In *Engineering and Managing Software Requirements*, chapter 4, pages 69–94. Springer, 2005.

[4] M. Cohn. *User stories applied: For agile software development.* Addison-Wesley Professional, 2004.

[5] M. Cohn and R. Martin. *Agile Estimating and Planning.* Prentice Hall, 2005.

[6] F. Fransella, R. Bell, and D. Bannister. *A Manual for Repertory Grid Technique.* John Wiley & Sons, Ltd., 2004.

[7] J. E. Hannay and H. C. Benestad. Perceived productivity threats in large agile development projects. In *Proc. 4th Int'l Symp.Empirical Software Engineering and Measurement (ESEM)*, pages 1–10. IEEE Computer Society, 2010.

[8] V. Heikkilä, A. Jadallah, K. Rautiainen, and G. Ruhe. Rigorous support for flexible planning of product releases—a stakeholder-centric approach and its initial evaluation. In *Proc. 43rd Hawaii Int'l Conf. System Sciences (HICSS 2010)*, pages 1–10. IEEE Computer Society, 2010.

[9] D. Jankowicz. *The Easy Guide to Repertory Grids.* John Wiley & Sons, Ltd., 2004.

[10] M. Jørgensen. A review of studies on expert estimation of software development effort. *J. Systems and Software*, 70(1–2):37–60, 2004.

[11] J. Karlsson and K. Ryan. A cost-value approach for prioritizing requirements. *IEEE Software*, 14(5):67–74, 1997.

[12] G. A. Kelly. *The Psychology of Personal Constructs.* Norton, 1955.

[13] K. Krippendorff. *Content Analysis: An Introduction to its Methodology.* Sage, second edition, 2004.

[14] L. Lehtola, M. Kauppinen, and S. Kujala. Requirements prioritization challenges in practice. In *Proc. Product-Focused Software Process Improvement (PROFES 2004)*, volume 3009 of *Lecture Notes in Computer Science*, pages 497–508. Springer, 2004.

[15] M. Li, M. Huang, F. Shu, and J. Li. A risk-driven method for eXtreme programming release planning. In *Proc. 28th Int'l Conf. Software Engineering (ICSE 2006)*, pages 423–430. IEEE Computer Society Press, 2006.

[16] S. Maurice, G. Ruhe, O. Saliu, and A. Ngo-The. Decision support for value-based software release planning. In *Value-Based Software Engineering*, pages 247–261. Springer, 2006.

[17] G. A. Moore. *Crossing the Chasm.* Harper Business, revised edition, 2002.

[18] A. Ngo-The and G. Ruhe. Optimized resource allocation for software release planning. *IEEE Trans. Software Eng.*, 35(1):109–123, 2009.

[19] A. Ngo-The, G. Ruhe, and S. Wei. Release planning under fuzzy effort constraints. In *Proc. 3rd IEEE Int'l Conf. Cognitive Informatics*, pages 168–175. IEEE Computer Society, 2004.

[20] A. Ngo-The and M. O. Saliu. Fuzzy structural dependency constraints in software release planning. In *Proc. 14h IEEE Int'l Conf. Fuzzy Systems*, pages 442–447. IEEE Computer Society Press, 2005.

[21] S. L. Pfleeger. Albert Einstein and empirical software engineering. *IEEE Computer*, 32(10):32–38, Oct. 1999.

[22] B. Regnell, M. Host, and R. B. Svensson. A quality performance model for cost-benefit analysis of non-functional requirements applied to the mobile handset domain. In *Proc. 13th Int'l Working Conf Requirements Engineering: Foundation for Software Quality (REFSQ 2007)*, volume 4542, pages 277–291, 2007.

[23] E. M. Rogers. *Diffusion of Innovations.* Free Press, fifth edition, 2003.

[24] G. Ruhe. *Product Release Planning Methods, Tools and Applications*, chapter 10, pages 201–224. Auerbach Publications, 2010.

[25] G. Ruhe. *Product Release Planning Methods, Tools and Applications*, chapter 7, pages 125–143. Auerbach Publications, 2010.

[26] G. Ruhe, A. Eberlein, and D. Pfahl. Trade-off analysis for requirements selection. *Int'l J. Software Engineering and Knowledge Engineering*, 13(4):345–366, 2003.

[27] G. Ruhe and J. Momoh. Strategic release planning and evaluation of operational feasibility. In *Proc. 38th Annual Hawaii Int'l Conf. System Sciences (HICSS 2005)*, volume 5, page 313. IEEE Computer Society, 2005.

[28] G. Ruhe and A. Ngo. Hybrid intelligence in software release planning. *Int'l J. Hybrid Intelligent Systems*, 1:99–110, 2004.

[29] T. L. Saaty. *Multicriteria Decision Making: The Analytic Hierarchy Process: Planning, Priority Setting, Resource Alloca-*

tion. RWS Publications, 1990.

[30] K. Schwaber. *Agile Project Management with Scrum*. Microsoft Press, 2004.

[31] M. Sliger and S. Broderick. *The Software Project Manager's Bridge to Agility*. Addison Wesley, 2008.

[32] M. Svahnberg, T. Gorschek, R. Feldt, R. Torkar, S. Bin Saleem, and M. U. Shafique. A systematic review on strategic release planning models. *Information and Software Technology*, 52:237–248, 2010.

[33] M. Van den Akker, S. Brinkkemper, G. Diepen, and J. Versendaal. Software product release planning through optimization and what-if analysis. *Information and Software Technology*, 50(1–2):101–111, 2008.