Full citation: Kirk, D., & MacDonell, S. (2009) A simulation framework to support software project (re)planning, in Proceedings of the 35th Euromicro Software Engineering and Advanced Applications (SEAA) Conference. Patras, Greece, IEEE Computer Society Press, pp.285-292. http://dx.doi.org/10.1109/SEAA.2009.64

A Simulation Framework to Support Software Project (Re)Planning

Diana Kirk, Stephen MacDonell

SERL, Auckland University of Technology
Private Bag 92006, Auckland 1142,
New Zealand
dkirk@aut.ac.nz, stephen.macdonell@aut.ac.nz

Abstract

Planning and replanning software projects involves selecting activities according to organisational policies, project goals and contexts, deciding how to effect the activities, and dealing with uncertainty in activity outputs. There is at the present time no general model to support project managers with all of these tasks. The contributions of this paper are to propose a set of properties that are desirable in a model for (re)planning and to create a framework based on these properties. The purpose of the framework is to support the modelling and simulation of (re)planning during software projects. Key aspects of the framework are a focus on project objectives as drivers of activity selection, and activity prediction that supports uncertainty and that may be based on previous activity data, expert opinion or experimental evidence. We present a 'proof-of-concept' case study to illustrate how the framework can be applied to support planning.

Keywords - Software system; simulation; planning; planning framework

1. INTRODUCTION

Planning for software projects involves selecting the process and management activities that will be carried out and making decisions about some of the parameters for these activities, for example, who will carry them out and which tools will be provided for support. Sometimes the activities are fixed according to which software lifecycle is mandated at a company level. In this case, planning involves parameter selection only. In other instances activities are selected in a flexible way according to project specifics.

As a project progresses, replanning occur. Again, in some cases parameters only are adjusted, for example, when more experienced developers are assigned to replace inexperienced ones on a project experiencing quality issues. In other cases, a change in the activity itself is indicated, for example, when a planned review is omitted for a project that is behind schedule.

Although a number of models exist to support (re)planning, most are limited to the selection of parameters i.e do not provide general support for activity selection and uncertainty. The contribution of this paper is to propose a framework that addresses these gaps.

Two key ideas are of interest in the (re)planning effort. The first is the idea that all process and management activities are aimed at moving the project towards some desired goals. For example, a delivered product generally must meet some agreed quality criteria and at an agreed cost. Replanning occurs when the project is not on track to meet goals and planned activities are adjusted or changed in an effort to increase the likelihood that the goals will be met. The second idea is that of prediction uncertainty. For example, a project manager might predict that a coding activity will consume a certain number of developer hours but this number is approximate and its accuracy reflects, among other things, the experience of the project manager. Such uncertainty is present even when predictions are based on company data for previous projects, as the source projects that provide the data are likely to differ in some way from the project under consideration. For example, it is unlikely that all developers on the new project have the same experience and subject area knowledge as those from earlier projects.

The contributions of this paper are to propose a set of properties that are desirable in a model for (re)planning and to create a framework based on these properties. The purpose of the framework is to support the modelling and simulation of (re)planning during software projects. Key aspects of the framework are a focus on project objectives as drivers of activity and parameter selection, and activity prediction that supports uncertainty and that may be based on previous activity data, expert opinion or experimental evidence. This means that the framework is appropriate for both organisations with previous process data and organisations with none.

In Section 2, we suggest some properties that are desirable in a model for (re)planning. In Section 3, we describe our proposed framework. In Section 4, we present a 'proof-of-concept' case study to illustrate practical use of the framework. In Section 5, we discuss

related work and, in Section 6, we discuss limitations of the proposed framework and plans for implementation and evaluation.

2. PROPERTIES FOR (RE)PLANNING

As a preliminary to framework creation, we consider the kinds of properties that are relevant for modelling the (re)planning function. The aim is to provide some context for our proposed framework and for the assessment of this and related work. To source the properties, we consider some orthogonal aspects of (re)planning and use each as a basis for property identification. The resulting list is a proposed set of necessary properties.

2.1 Organisational context

The first set of properties relates to the project within the larger organisational context.

• *Focus on objectives.* Software projects exist to meet specific organisational objectives. These objectives may differ between projects. For example, one project may be delivering software to an early adopter market while another may be creating a high dependability system. Key factors for the former may not include 'reliability', while those for the latter certainly will. Planning must aim to achieve a successful outcome for the key factors identified for each project.

• *Product line management.* A project may be only one of many in the development or maintenance of a product line. For many projects, then, the state of the 'project system' is not empty at commencement. The delivered product will be associated with a 'cost so far', perhaps a certain level of 'reliability', and the product source will be characterised by a certain 'complexity'. Project personnel also have a state prior to project commencement, for example, a certain level of 'experience' will be associated with project staff. These will all affect project decisions, for example, 'include refactoring to reduce source complexity' or 'omit inspections to reduce cost'.

2.2 Process activities

The next set of properties relates to activity selection. We include any activities required to meet the defined organisational objectives.

• *System perspective.* Several authors warn against a focus on single outcomes as this leads to the identification of local project maxima only i.e. to a sub-optimisation of the whole system [1], [2], [3]. Generally, several objectives are of interest and these may relate to any aspect of the system, for example, to people and project as well as product. For example, a project manager will most likely need to balance a reduction in defects as a result of a new inspection technique with the cost of implementing such a technique and may also want to monitor the motivation of project team members.

• *Process granularity.* There is generally a need to plan at the 'whole process' level (for example, as for most cost estimation models) but support for planning at the task level (for example, 'code from designs') is also desirable.

• *Process content.* As noted above, objectives may include those other than product-related, for example, 'increase developer product knowledge'. This means that we would like to model 'normal' activities, such as 'design' or 'test', and also activities such as 'team meeting' and 'knowledge transfer'.

• *Process comparison.* Planning and replanning activity often includes some 'what-if' analysis. For example, how much difference will it make to final outcomes if a formal inspection is replaced by a less formal inspection carried out by an experienced developer?

2.3 Process management

The final set of properties relates to managing the process.

• *Policy support.* Project managers must often decide when to commence or complete a development activity, for example, when to start coding or stop testing. The rule of 'start coding when designs are complete' is overly simplistic as in the real world there is generally some overlap. These kinds of decisions will depend upon context and a useful framework must support them but not dictate policy i.e. we desire a decoupling of process and policy.

• *Process monitoring.* Project managers generally monitor projects by checking the status of activities with project staff. Sometimes this results in a change to an activity-in-progress, for example, to add an experienced developer to an activity that is late. An effective framework must support this 'mid-activity' monitoring and replanning function.

• *Planning uncertainty.* Software projects are characterised by uncertainty [4], [5] and so expected values for outcomes and predictions for activities are more accurately characterised as distributions rather than single points [6], [7], [8], [9].

All properties presented above are relevant for (re)planning and decision-making during software projects. At this stage, we suggest that the properties represent a necessary set but we do not claim completeness or orthogonality. For example, there may be other important properties not yet identified, and it is likely that 'Product Line Management' and 'Focus on Objectives' are related.

In the next Section, we propose a framework to support (re)planning based on the above properties.

3. PROPOSED FRAMEWORK

3.1 System Variables-of-Interest (SVoI) as drivers of activities

The approach to planning the software process for a project generally involves selecting which life-cycle model or methodology will be applied, how this will be tailored for the project and who will carry out the chosen activities. For example, an 'agile' group may implement *eXtreme Programming (XP)*, and tailoring discussions may centre around how much documentation to produce.

Groups favouring a more traditional life-cycle may implement an 'iterative waterfall' with tailoring discussions relating to the process for inspections.

We have suggested elsewhere that a change in focus from *performing activities* to *transforming System Variables-of-Interest (SVoI)* provides a more holistic and flexible support for decision-makers [10], [11]. In this approach, we first identify the key objectives for the project, for example 'quality', and establish each as a *system factor-of-interest*. Each key factor is then operationalised as at least one *SVoI* and assigned an appropriate target value, for example, 'no more than ten known defects'.

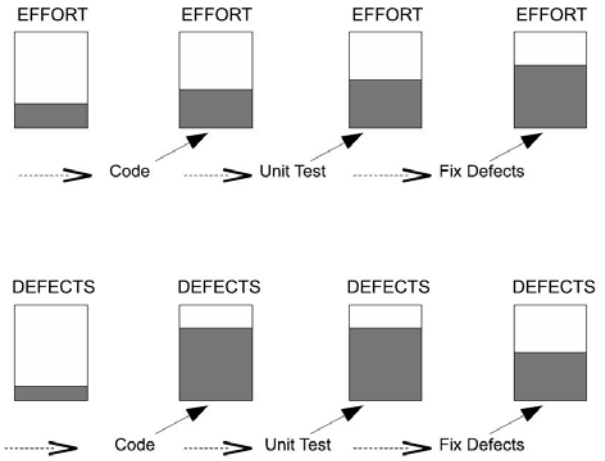

Figure 1. Activities changing Effort and Defects

Process and management activities are then viewed as transformations on these variables. The aim of transformations is to 'move' the values of the variables towards the desired outcome values. As illustration, in Figure 1, we depict three activities, 'Code', 'Unit test' and 'Fix defects', changing system variables representing 'Effort' and 'Defects'.

In this illustration, the 'Code' activity increases 'Effort' and a number of 'Defects' are injected. 'Unit Test' also increases 'Effort' but no change is effected to 'Defects' (although some defects may be uncovered). 'Fix Defects' increments 'Effort' and reduces 'Defects' as existing defects are resolved and a smaller number are injected as a result of the activity.

Note that we are transforming variables, not process outputs, i.e. documents. In our approach, documents may represent variables but the variables are, in fact, abstractions of process outcomes. For example, the quality-related aspects of the process may be captured in a number of documents, but the SVoI for the project represents an abstraction of quality in which we have some interest i.e. represents how we operationalise quality.

3.2 RealisedProcess

The approach described above supports the creation of RealisedProcesses to represent variable transformations [11]. The standard use of the term *process* generally refers to a description of a set of technical tasks and does not include any non-technical factors, for example, relating to humans. Our definition as *transformation on SVoI* means that all aspects of the transformation are included. This means that we can include, for example, factors such as 'developer experience' and 'user satisfaction'. If we consider an inspection that transforms 'Defects' and 'Effort', we understand, for example, that two actual inspections will effect different sizes of transformations according to the experience of the participants and that the experience values may also increase as a result of participation in the inspection.

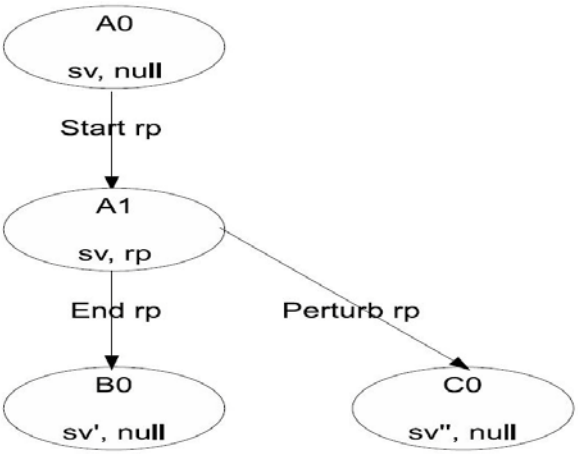

Figure 2. State machine for RealisedProcess

In order to simulate a RealisedProcess, we require a more formal representation. In Figure 2, we show a state machine for RealisedProcess *rp* with nodes representing system states and edges transitions between states. State transition may occur in a planned way, for example, when an activity is completed as planned, or in an unplanned way, for example, when a project manager perturbs the activity mid-way. The state space is the vector of SVoI for *rp* and we denote this as *sv*. Points of visibility into *rp* are the input stimulii. These are *Start rp*, *Perturb rp* and *End rp*.

Prior to commencement, the system is in state *A0* i.e. the system variables vector has state *sv* and no activity is active (*rp* is 'null'). Once *rp* is commenced, the system moves to state *(*A1), where *rp* has state *rp*. Note that in A1 the state of *sv* is unchanged because there is no visibility into the system while an activity is in progress, unless the activity is perturbed, taking the state to *C0*. Transformation from *A1* to *B0* moves the state of *sv* to the new value *sv'* and returns the activity state to 'null'. The *accepting states* for *rp* are states for which values of system variables are compliant with objectives. More details are available in [11].

3.3 Distributions for SVoI and transformations

Software projects are characterised by many kinds of uncertainty. Some relate to issues of vagueness and ambiguity and include, for example, failure to clearly define objectives, lack of a clear specification and product complexity [4]. Others relate to human aspects and include factors such as the motivation, availability and capabilities of the project participants and shallow subject area knowledge [4]. Still others relate to the project environment and include factors such as dependence on external participants [5] and market change [9].

One problem resulting from this inherent uncertainty is that project planners cannot reliably predict outcomes, for example, relating to cost and quality, in a deterministic way. A common approach to this problem is to represent predictions by a prediction interval along with confidence level [12], [7]. This approach allows 'best case' and 'worst case' scenarios to be explored. Another approach involves representing predictions as probability distributions and applying a stochastic approach to model the project [6], [7], [8]. In this approach, outcomes are also represented as probability distributions.

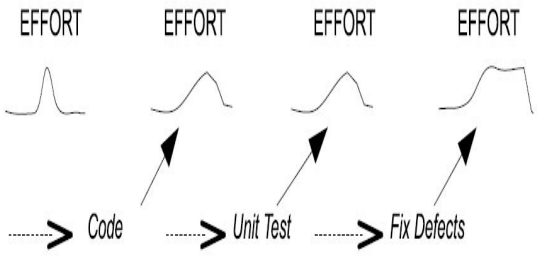

Figure 3. Activities changing Effort with uncertainty

For our framework, we represent a prediction of how an activity changes a SVoI as a probability distribution. In Figure 3, we again depict transformations on 'Effort' when activities 'Code', 'Unit test', and 'Fix defects' are carried out.

The end result for the system variable 'Effort' is a distribution that effectively represents the risk inherent in the prediction [6]. For example, a distribution skewed to the right alerts project management to the need to plan for higher levels of required effort [6]. Predictions supplied may be based on existing prior project data, expert opinion or evidence obtained from studies and formal experiments. Simulation occurs by a stochastic execution of the state machine described in the previous Section and the elements of the state vector *sv* are now represented as distributions rather than point values. We note that the choice of probability distributions for predictions provides us with flexibility, in that all of point values (single value), intervals (rectangular), worst-case (triangular), mean plus standard deviation (normal) and activity-specific (custom curve, as suggested by Kitchenham et. al. [7]) are supported.

3.4 Framework and (re)planning properties

We submit that the framework presented above effectively addresses all (re)planning properties described in Section 2.

1) Organisational context:

• *Focus on objectives.* Objectives are defined as 'system factors of interest' and operationalised as SVoI.

• *Product line management.* SVoI have a specific state at project start as a result of previous projects.

2) Process activities:

• *System perspective.* Choice of objectives is unconstrained and multiple objectives may be considered.

• *RealisedProcess granularity.* There are no constraints on the transformation size i.e. granularity is unconstrained.

• *RealisedProcess content.* There are no constraints on the selection of objectives or transformation content.

• *RealisedProcess comparison.* This is a straightforward comparison of the effects of transformations. We note that comparison is possible only if the compared RealisedProcesses transform the same SVoI [11].

3) Process management:

• *Policy support.* System state is described by values of SVoI and activities may be defined to commence when these variables reach specific values. For example, a 'code' activity may commence when a 'design' state is at 80 percent or 100 percent. Policy and process are decoupled.

• *RealisedProcess monitoring.* The state that describes an active activity accepts a 'Perturb rp' event which returns the activity value to 'null' and changes the SVoI to the values at the time of perturbation.

• *Planning uncertainty.* Variables-of-interest and transformations are represented as probability distributions.

4. PROOF-OF-CONCEPT CASE STUDY

In this Section, we present a proof-of-concept case study to illustrate how the framework may be applied to support project management planning. For ease of illustration, we select a study that does not involve uncertainty. Data for the study is taken from a study by MacCormack et. al. [13].

A project manager would like to increase customer satisfaction levels (*objective*) and, after interviewing some key customers, learns that quality is an issue. He is also interested in maintaining project expenditure at existing levels (*objective*). He decides to focus on increasing quality while maintaining cost levels (*factors-of-interest*). He finds a study in the literature that suggest that adopting a practice of integration or regression testing at code check-in is associated with a 36 percent reduction in defect rates and the introduction of design reviews is correlated with a defect rate reduction of 55 percent [13]. The organisation maintains a data repository for past projects and data includes effort and defect counts. He decides to focus on these for his investigation (*SVoI*). He understands from the repository that the expected effort for 'design, code and build' for his project is 2,000 person hours, duration is 25 weeks and the expected final defect count is 60. Developers advise that regression testing will incur a cost of 2 person hours per run. As the project currently carries out weekly builds, this relates to an additional cost of 50 person hours. Developers also advise that design reviews will add an overhead of 100 person hours. From the study, the manager learns that the introduction of daily builds is correlated with a 35 percent increase in productivity. He applies the framework to explore the trade-offs between expected productivity and defect levels when the various options are implemented.

For this exploration, our SVoI are 'Effort' and 'Defects'. We illustrate the various options in Figure 4. Starting values for 'Effort' and 'Defects' will be dependent on previous activities, but will be the same for all options. For simplicity, we assume zero starting values for both 'Effort' and 'Defects'. From the illustrations, we see that the choice between 'Regression Testing' and 'Design Reviews' is not clear cut. Although 'Design Reviews' provides a better outcome as regards defect totals, the expected effort is greater. The introduction of 'Daily Builds' alone reduces 'as-is' effort to 1,300 person hours. However, when combined with 'Regression Testing', the regression testing overhead is now 250 person hours, and so the option 'Daily Builds plus Regression Testing' has an expected effort of 1,550, while the 'Daily Builds plus Design Reviews' has expected effort 1,400. The manager realises that the objectives relating to quality and project expenditure can be met most effectively by introducing a regime of daily builds to control effort plus design reviews to increase quality.

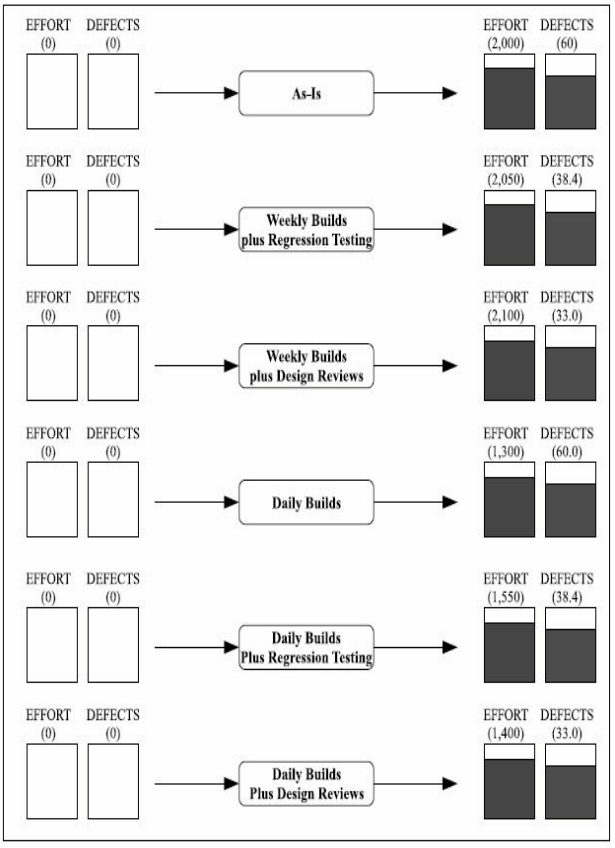

Figure 4. Exploring trade-offs

5. RELATED WORK

In this section, we overview some research aimed at supporting planning during software projects. We then compare models with our proposed framework using the desired (re)planning properties described above as criteria.

5.1 Modelling and simulation

The main source of related work is the modelling and simulation community. Software process simulation and modelling has become an "increasingly active research area" with growing numbers of publications and related activities [14]. Techniques applied include *discrete-event simulation* and *system dynamics*.

In a *discrete event simulation*, discrete entities ('units of traffic') move ('flow') from point to point in the system. Entities instigate and respond to events (things that happen and change the state of the system). When this paradigm is applied to software development, the 'product artifact' entities flow through process blocks. Reported limitations of this paradigm include an inability to model smoothly-varying aspects, for example, 'schedule pressure', and the need to pre-define activities [15] i.e. a direct application of this paradigm would present difficulties if we want to change the process in a non predetermined way.

In a *system dynamics* approach, a process is treated as a system with many feedback loops. System variables are represented as *levels* and these levels rise and fall according to *flows* created as a result of the effects of other 'levels'. Feedback from individual flows is linear, and the total result for a level may be exponential increase, exponential decrease or oscillations depending upon the multiplication factors for the various flows. Variables thus change in a 'continuous' way and the approach is suitable for modelling, for example, developer motivation changing through long projects. One limitation of the approach is the inability to capture attributes for variables represented as 'levels', for example, the attribute 'code complexity' for level 'amount of completed code' [16]. A second limitation is the need to change the model if the underlying process changes.

The specific models overviewed below address the issue of flexibility by building processes from a number of predefined activity 'building blocks'.

Lakey [2] introduces a model to support software project prediction and management. The model is intended as a theoretical framework. It comprises four building blocks, 'preliminary design', 'detailed design', 'code and unit test' and 'subsystem integration and test'. In this framework, project-specific process models are built by creating an appropriate number of building blocks and calibrating the equations for each with project, process and product data from the project to be modelled. Examples of project factors included are 'communication overhead', 'tool support' and 'skill levels'. Examples of process factors are 'defects injected' and 'estimated calendar weeks'. Product factors include 'size' and 'quality'. A strength of this framework is the inclusion of cost, schedule and quality performance parameters in a holistic system as "the primary software project performance parameters of cost, schedule and quality are not independent, and they cannot be tracked and managed independently". However, customisation is achieved by copying and renaming building blocks to achieve the correct process structure and then providing the relevant input values. This means that there is no possibility of representing any activities that do not comply with one of these blocks. We suggest that customisation thus refers to changing input values rather than changing the process. Another limitation is in the pre-definition of the factors

that are believed to affect outcomes. The beliefs are effectively model assumptions.

Munch applies a patterns approach to the development of custom-tailored process models [17]. He believes that "The development of high-quality software or software-intensive systems requires custom-tailored process models that fit the organizational and project goals as well as the independent contexts" (page 1). In Munch's solution, a process pattern is a reusable fragment of a process model that represents an activity. Patterns can be combined to represent combinations of process models. Information for each pattern includes attributes and a description of how attributes change when the pattern is applied, for example, causing a change to 'reliability' [17]. In this model attributes may relate to process state (for example, 'not in maintenance activity') or process goals (for example, 'Maximal effort is less than 2000'). Required goals are thus modelled as restrictions on project attributes and include only those over which the project has control. This means that the model does not support SVoI such as 'developer subject area knowledge' and other human-related goals i.e. the model is fixed as regards inclusion of SVoIs. In addition, the rules for transformation form an integral part of the model i.e. assumptions are embedded in the model.

Raffo et. al. describe an approach for creating *Generalised Process Simulation Models (GPSM)* [18]. The approach consists of constructing a process from a library of generic process building blocks, for example, relating to 'Design', configuring the inputs to blocks for specific environments and viewing outputs relating to time, cost, quality and functionality. Although the approach supports a degree of flexibility in process construction, there is an assumption of 'traditional process' and a restriction of outputs to those relating to time, cost, quality or functionality [19]. This means that the GPSM model as constructed cannot be used for simulating less traditional processes or for modelling, for example, the effects of 'team meeting' on 'developer product knowledge'. The lack of a systems perspective also means that product line development is not supported.

5.2 Other frameworks

Several authors have proposed approaches that reduce risk by enabling a planner to select activities that will support organisational objectives. Models such as *Spiral* [20] and *Rational Unified Process (RUP)* [21] aim to address risk by facilitating flexibility as regards which development activities are performed. However, 'performing activities' is the focus of interest for each and there is no concept of an activity being defined by how it transforms SVoI.

Recent contributions from collaborations involving the University of Southern California include combining process elements [22], tailoring the process according to business cases [23], [9] and dealing with uncertainty by fixing the system variable 'Schedule' [9]. The underlying paradigm for these contributions is that of *value-based engineering*, where key mechanisms include understanding what is the key objective for a project from a value perspective (for example, cost, quality), selecting activities that will ensure the objective is reached in the most cost-effective way and monitoring the project to ensure both objective and activity selection remain appropriate. The modelling of objectives is not formalised and so the contributions support only a subset of the points described above. We do, however, note that the approach described in this paper supports the ideas of value-based software engineering [24] as the SVoI represent relevant values for a project and these can be monitored as the project progresses.

	Focus on objectives	Product line	System perspective	Process granularity	Process content	Process comparison	Policy support	Process monitoring	Planning uncertainty
Lakey 2003	√	X	X	X	X	√	X	X	X
Munch 2005	√	X	X	√	X	X	X	X	X
Raffo 2007	√	X	X	√	X	√	X	X	X
Basili&Rombach 1987	√	X	X	√	X	X	X	X	X
Rao et. al. 2008	√	√	X	√	X	√	X	X	√
USC	√	X	X	√	X	X	X	√	√

Figure 5. Comparison of various planning modelling approaches

Other tailoring approaches include Basili and Rombach's approach for tailoring processes towards project goals and environments [25]. Again, a specific project objective is identified and activities selected that will ensure the objective is met. However, there is no provision for examining multiple project objectives and the framework upon which the approach is based contains a number of process-related assumptions that constrain flexibility.

5.3 Practical application

Rao et. al. report a successful implementation of an initiative to create a framework for quantitative project management [8]. Predictions relating to effort, quality, schedule and scope are sourced from company baseline capability reports and comprise distributions for each planned activity. The expected results from a project are obtained by applying a Monte Carlo simulation technique. Results are also distributions and are effectively "a function of all the distributions associated with each activity" [8]. Activities are selected to achieve results that best meet objectives.

The described implementation provides an excellent example of a constrained implementation of the framework proposed in this paper. Constraints include the limiting of project objectives to those relating to effort, quality, schedule and scope and the assumption of normality for activity input distributions. The first means that objectives such as those relating to humans, for example, 'motivation', and economic value, are not supported. The assumption of normality is not consistent with the possibility that transformations relating to some activities may be better described by, for example, a Gamma distribution [7].

5.4 Comparison

In Figure 5, we compare the schemes discussed in this section with respect to the desired (re)planning properties presented in Section II.

It is evident that none of the models considered here supports a 'system perspective' i.e. objectives are either pre-defined or constrained. The lack of extensibility in objectives means that flexibility in 'process content' is not supported, for example, 'team meeting' cannot be modelled. Indeed, none of the models supports the introduction of new kinds of activity, for example, 'test first design' or 'pair programming'. We also observe that decoupling of process and policy is not supported i.e. it is not possible to model, for example, 'commence coding when designs are 80 percent complete'.

6. DISCUSSION AND FUTURE WORK

We have proposed a set of properties for models for software project (re)planning and a framework to support (re)planning based on these properties. One aspect of the framework, the provision of a distribution to describe the effect of a transformation on a system variable, has both practical advantages and disadvantages. A distribution effectively 'wraps up' all factors that might affect outcomes as a single curve. This means that, at the present time, when there is little evidence to support ideas about what are the key influencing factors, planners can supply curves based on their experience i.e. simulations can still take place without assumptions being embedded in the model itself. In cases where either suitable data or evidence does exist, the curves will be provided from the data or evidence. The source of the input distributions is irrelevant for the framework. One limitation of the use of distributions relates to the need for planners to supply mathematical distributions for each transformation and for each SVoI. To facilitate this, we are researching the application of a 'fuzzy' layer to provide a more friendly interface to planners while providing a distribution to the simulation engine. Another possible limitation relates to issues of sensitivity - it is not clear whether long process chains will be overly sensitive to the shape of curves early in the chain. This is an area for future research.

Other immediate plans for the research are to further investigate the proposed properties with a view to understanding conditions of necessity and sufficiency and to implement the framework as a basis for studies within the local community of small to medium sized software organisations.

7. REFERENCES

[1] B. A. Kitchenham, S. L. Pfleeger, D. C. Hoaglin, K. E. Emam, and J. Rosenberg, "Preliminary Guidelines for Empirical Research in Software Engineering," *IEEE Transactions on Software Engineering*, vol. 28, no. 8, 2002.

[2] P. B. Lakey, "A Hybrid Software Process Simulation Model for Project Management," in *Proceedings of the 2003 International Workshop on Software Process Simulation and Modeling (ProSim'03)*, Portland, Oregan, U.S.A., 2003.

[3] M. Lehman, "Process Modelling - Where Next," in *Proceedings of the 1997 Conference on Software Engineering*. IEEE Computer Society Press, 1997.

[4] R. Atkinson, L. Crawford, and S. Ward, "Fundamental uncertainties in projects and the scope of project management," *International Journal of Project Management*, vol. 24, pp. 687–698, 2006.

[5] O. Perminova, M. Gustaffson, and K. Wikstrom, "Defining uncertainty in projects a new perspective," *International Journal of Project Management*, vol. 26, pp. 73–79, 2007.

[6] A. Connor, "Probabilistic estimation of software project duration," *New Zealand Journal of Applied Computing and Information Technology*, vol. 11, no. 1, pp. 11–22, 2007.

[7] B. Kitchenham and S. Linkman, "Estimates, Uncertainty and Risk," *IEEE Software*, vol. May/June, 1997.

[8] U. S. Rao, S. Kestur, and C. Pradhan, "Stochastic Optimization and Modeling and Quantitative Project Management," *IEEE Software*, vol. May/June, pp. 29–36, 2008.

[9] D. Yang, B. Boehm, Y. Yang, Q. Wang, and M. Li, "Coping with the Cone of Uncertainty: An Empirical Study of the SAIV Process Model," in *ICSP 2007, Lecture Notes in Computer Science (LNCS)*, Q. Wang, D. Pfahl, and D. Raffo, Eds., vol. 4470. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 37–48.

[10] D. Kirk and E. Tempero, "A Conceptual Model of the Software Development Process," in *Proceedings of the 6th International Workshop on Software Process Simulation and Modeling (ProSim'05)*. St. Louis, Missouri: Fraunhofer IRB, 2005.

[11] D. Kirk, "A flexible software process model," Ph.D. dissertation, University of Auckland, Auckland, New Zealand, 2007.

[12] M. Jorgensen, K. H. Teigen, and K. Molokken, "Better sure than safe? Over-confidence in judgement based software development effort prediction intervals," *Journal of Systems and Software*, vol. 70, pp. 79–93, 2004.

[13] A. MacCormack, C. Kemerer, Cusumano, and Crandall, "Trade-offs between Productivity and Quality in Selecting Software Development Practices," *IEEE Software*, vol. 20, no. 5, pp. 86–93, 2003.

[14] H. Zhang, B. Kitchenham, and D. Pfahl, "Reflections on 10 Years of Software Process Simulation Modeling: A Systematic Review," in *ICSP 2008, Lecture Notes in Computer Science (LNCS)*, Q. Wang, D. Pfahl, and D. Raffo, Eds., vol. 5007. Berlin, Heidelberg: Springer-Verlag, 2008, pp.345–356.

[15] A. Drappa and J. Ludewig, "Quantitative modeling for the interaction simulation of software projects," *Journal of Systems and Software*, vol. 46, no. 2/3, 1999.

[16] R. H. Martin and D. M. Raffo, "Application of a hybrid process simulation model to a software development project," *Journal of Systems and Software*, vol. 59, no. 3/3, 2001.

[17] J. Munch, "Goal-oriented Composition of Software Process Patterns," in *Proceedings of the 6th International Workshop on Software Process Simulation and Modeling (ProSim'05)*. St. Louis, Missouri: Fraunhofer IRB, 2005, pp. 164–168.

[18] D. Raffo, U. Nayak, and W. Wakeland, "Implementing Generalized Process Simulation Models," in *Proceedings of the 6th International Workshop on Software Process Simulation and Modeling (ProSim'05)*. St. Louis, Missouri: Fraunhofer IRB, 2005, pp. 139–143.

[19] D. M. Raffo, “System and method for simulating product design and development,” http://www.freepatentsonline.com/20050160103.html, July 2005.

[20] B. W. Boehm, “A Spiral Model of Software Development and Enhancement,” *IEEE Computer*, vol. May, no. 11, 1988.

[21] P. Kruchten, *The Rational Unified Process: An Introduction, Second Edition*. United States of America: Addison-Wesley, 2000.

[22] J. Bhuta, B. Boehm, and S. Meyers, “Process Elements: Components of Software Process Architectures,” in *SPW 2005, Lecture Notes in Computer Science (LNCS)*, M. Li, B. Boehm, and L. Osterweil, Eds., vol. 3840. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 332–346.

[23] L. Huang, H. Hu, J. Ge, B. Boehm, and J. Lu, “Tailor the Value-Based Software Quality Achievement Process to Project Business Case,” in *SPW/ProSim 2006, Lecture Notes in Computer Science (LNCS)*, Q. W. et. al., Ed., vol. 3966. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 56– 63.

[24] B. Boehm and L. G. Huang, “Value-Based Software Engineering: A Case Study,” *IEEE Computer*, vol. 36, no. 3, pp. 21–29, 2003.

[25] V. R. Basili and H. D. Rombach, “Tailoring the Software Process to Project Goals and Environments,” in *Proceedings of the Ninth International Conference on Software Engineering*, IEEE. IEEE Computer Society Press, 1987.