Dynamic project performance estimation by combining static estimation models with system dynamics

KeungSik Choi *, Doo-Hwan Bae

Department of EECS, Korea Advanced Institute of Science and Technology (KAIST), 373-1 Guseong-dong, Yuseong-gu, Daejon 305-701, Republic of Korea

Article history:
Received 5 May 2007
Received in revised form 29 February 2008
Accepted 2 March 2008
Available online 18 March 2008

Keywords:
Dynamic software process simulation
System dynamics
Estimation model integration
Dynamic project performance estimation
Expert judgment technique

ABSTRACT

Changes in user requirements or project personnel occur frequently during project execution particularly in long-term and large-size projects. We need a tool which can estimate the effects of changing conditions to effectively manage the project.

This paper proposes a simulation method for dynamic project performance in terms of effort, schedule, and defect density changes in a dynamic project environment by combining COCOMO II with system dynamics. We apply expert judgment technique to overcome the lack of empirical data on the effects of dynamic project environment. We develop a simulation tool (available on the authors’ website) which has model adjustment parameters to reflect experts’ estimation on project characteristics. The simulation experiment on a military application development project demonstrates that the developed model can show the behavioral characteristics of a project suffering unanticipated and uncontrolled requirements creep. This helps project managers understand interactions between project factors and proactively evaluate and control the effects of dynamic project environment.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

Changes in user requirements, project personnel, or objectives occur frequently during project execution particularly in long-term and large-size government projects such as a military command and control system development. These changes force projects to deviate from their initial project plan (expected effort, milestone, etc.) and often cause them to fail. The unanticipated or uncontrolled project changes interactively influence many other project factors such as developer’s morale and make it difficult to manage the projects. Unless such changes are avoidable, we need to manage them by evaluating the impact of project environment changes and taking actions proactively.

This paper proposes a simulation method for dynamic project performance in terms of effort, schedule, and defect density changes in a dynamic project environment by combining COCOMO II with system dynamics. We utilize the estimation capability of a parametric estimation model which provides a statistical estimation based on a function of multiple variables, calibrated with regression with historical data [1]. Specifically, we use COCOMO II in our research because it is most popular and provides an externally and internally open model, enabling users to fully understand and deal with its inputs, outputs, internal models, and assumption [2].

We combine COCOMO II with system dynamics as follows: First, we derive a small-time incremented development rate (e.g., KSLOC or Function Point (FP)/day) of each phase to bring dynamics to COCOMO II by using the effort and schedule distribution data. Second, we integrate the effort, schedule, and defect density estimation models to analyze the trade-offs among them. Finally, we incorporate additional project factors to represent the effects of the dynamically changing project environment. The numerical relationships caused by the additional project factors are derived by using expert judgment technique, Delphi [3], which is useful in the absence of empirical project data.

We implement a dynamic project performance estimation tool using iThink software [4] and perform a simulation experiment on a military application development project. The parameter values of the simulation model are set based on authors’ experience and experts’ survey. The simulation results show that the model can reproduce the behavioral characteristics of a military project suffering a lot of creeping requirements. We do not insist that our model can exactly estimate the effects of changing project conditions, but argue that our model can help project managers better understand interactions between project factors and proactively control the long-term and large-size projects. At least, it can show the most significant cause–effect relationships and typically observed behavior patterns of a target domain.

In the next section, we provide background on COCOMO II and system dynamics. In Section 3, we introduce the researches which applied COCOMO to the dynamic software process simulation and compare them with our approach. In Section 4, we discuss the basis of using the drivers of COCOMO II in a dynamic simulation and present how to combine static estimation models of COCOMO II with system
dynamics. In Section 5, we show a simulation experiment on a military application development project and discuss the model verification and validation. In Section 6, we conclude and discuss future work.

2. Background

2.1. Estimation models of COCOMO II

The COCOMO suite includes several extensions, each of which is an independently working model, to help project managers to reason about the development effort, schedule, and quality (residual defects of a system) of a project. COCOMO enables us to represent the characteristics and domain of a software development project using cost drivers and scale factors.

2.1.1. Effort and schedule estimation models

The expressions in (1) show the COCOMO II effort and schedule estimation models [2], where $A$ is a productivity constant. The inputs of COCOMO II are the size of software development, scale exponent ($E$), and effort multipliers ($EM$). The scale exponent is an aggregation of five scale factors, which can represent normative behavioral characteristics of different domains such as a commercial, system, or military domain. The effort multipliers are the product of cost drivers, which are used to capture characteristics of the software development that affect the effort to complete the project. COCOMO II outputs the effort in Person–months (PM) and the development schedule (TDEV) in months.

$$\text{PM} = A \times \text{Size}^E \times \prod_{i=1}^{n} EM_i$$

$$\text{TDEV} = 3.67 \times \text{PM}^{0.28 \times 0.2 - (E - 0.91)}$$

2.1.2. Quality estimation model: COQUALMO

COQUALMO (Constructive Quality Model), an extension of COCOMO II, estimates the quality of the software development in terms of the defect density (defects/LOC) [2,5–9]. It consists of Defect Introduction (DI) and Defect Removal (DR) submodels. For the purpose of COQUALMO, defects are classified based on their origins as requirements defects, design defects, and coding defects. DI drivers, all the scale factors and cost drivers of COCOMO II except the FLEX (Development Flexibility), determine the number of defects introduced. Each of the ratings of the DI drivers has the assigned numerical values.

The DR model is a post-processor to the DI model and is formulated by classifying defect removal activities into three orthogonal profiles namely automated analysis, people reviews, and execution testing and tools. Each of these profiles determines the Defect Removal Fraction (DRF). The DRF represents the numerical values of how much fraction of the requirements, design, and coding defects introduced by the DI model are removed.

Chulani [6] provides the expert-determined mapping table from DI drivers to the six levels of each of the three DR profiles. The organization’s process maturity (PMAT) and required software reliability (RELY) are the most important factors to determine the level of the DR profile. For example, when RELY and PMAT are both Nominal, then all three DR profiles are set at Nominal (i.e., DRF is 0.74), which means that approximately 74% of the defects are removed. The DI and DR submodels of COQUALMO are validated through case studies [2,5,6]. For more application examples, read the paper [8] and download the iDAVE tool [9].

2.2. System dynamics

System dynamics views the state variables of a system change in a time-continuous fashion and the behavior of a system arises from its feedback loop structure [10]. The dynamic software process simulation model quantitatively represents and evaluates the behavioral characteristics of the software development process using system dynamics. The dynamic model views processes as sets of activities or subsystems that influence one another, representing them using a set of coupled first order differential equations. Then the model represents the dynamic behavior of the processes as the model’s output by integrating the equations in discrete time steps [1].

Many researchers [11–17] have developed the dynamic project performance estimation models for understanding, planning, managing, and improving software development processes [18].

3. Related work

There have been many researches applying system dynamics and COCOMO in software process studies. However, most of the approaches use COCOMO to obtain the baseline performance of the project. For example, Abdel-Hamid and Madnick [19] use COCOMO to obtain the initial estimates for the project completion time and required effort, providing the baseline performance of the process. The model, however, is not applicable to other domains such as large-size military application development projects, because the primary focus of the model is that of medium-sized projects (16 K to 64 K LOC (Lines Of Code)) [19] and the model does not have variables which can represent characteristics of other domains.

Madachy introduced the Dynamic COCOMO, which is an extension based on the cost parameters varying over time versus traditional static assumptions [2]. He applied the concept of Dynamic COCOMO to a spiral life-cycle model to estimate the cost and schedule for multiple increments [20]. It considers changes due to the external volatility and feedback from user-driven change requests, and dynamically re-estimates and allocates resources in response to the volatility by re-calculating the effort estimation equation of COCOMO II. The model dynamically distributes the effort based on the development progress compared to the development schedule using the Rayleigh staffing curve [21].

Madachy’s approach gives us insights into the possibility and basis of using COCOMO II in dynamic software process simulation. However, it has some limitations for our research. First of all, the purpose of the Madachy’s model is different from our’s, so we cannot apply his model to our experiment. Second, the internal mechanism of the model is not publicly opened, which makes us difficult to use his approach. Third, the dynamics of his approach is mainly dependent on Rayleigh staffing curve, which requires additional calibration to apply it to different domains. Calibrating Rayleigh staffing curve requires organizations to have enough historical project data, however we want a modeling method which can be applied to organizations having not enough project data. Our approach can be an alternative measure for organizations having not enough project data. Finally, it does not incorporate project factors which are not covered by COCOMO II. The interactions among various project factors such as stakeholder’s friction and developer’s morale caused by dynamic project environment changes are very important in large-sized long-term projects. All of these reasons make us difficult to apply Dynamic COCOMO as it is.

Boehm and Huang discussed how to re-estimate a project using COCOMO II [22]. When unanticipated and often unnoticed changes are occur, corrective actions on the project plan need to be taken. This is an indication that the COCOMO II model needs to be recalibrated or extended to better fit the changing conditions. In such cases, one can re-calibrate the COCOMO II model parameters and feed the resulting revised estimates forward into the project’s milestone plans [22].
We extend their re-estimating concept by incorporating system dynamics which reflects the influence of time-continuously interacting project factors. The re-estimation of their approach estimates a project as if it is a new one, so the progress of the project at that moment is ignored and it shows only the end state of the project. However, our simulation tool can show the project’s progress continuously and enables project managers to perform “what-if” experiments.

4. Combining static estimation models with system dynamics

4.1. Basis of using the drivers of COCOMO II in a dynamic simulation

As we have discussed in Section 3, there is an approach which uses the drivers and driver values of the COCOMO II model in a dynamic environment. Madachy, one of the co-authors of COCOMO II [2], proposed Dynamic COCOMO which implements COCOMO with varying degrees of dynamic assumptions (e.g., increasing the personnel experience factors by learning). Boehm also argued the importance of re-estimating a project using COCOMO II to better fit the changing conditions [22].

We introduce time into the equations of the COCOMO II models to use drivers of them. We bring dynamics to COCOMO II by quantizing the equations using small-enough constant-time intervals. We approximate the functionalities being developed during a small-time interval as a continuous stream. It can be requirements specifications in requirements phase and codes in coding phase. Then we derive a development rate (e.g., job size/day) which is dynamically re-calculated by updating inputs of estimation models.

4.2. Basic mechanism of dynamic project performance estimation

Dynamic project performance estimation means that estimating a project’s performance in terms of expected effort, schedule, and defect density when the project conditions such as personnel, functionality (user requirements), deadline, etc. are dynamically changing during project execution.

Fig. 1 shows how the COCOMO II models such as effort, schedule, quality estimation models are combined with system dynamics. The project driving factors in Fig. 1 address the characteristics of a project and lead the project performance estimation. They include the functionality (expressed as job size in COCOMO II), scale factors (precededness, development flexibility, etc.), and cost drivers (product factors, platform factors, personnel factors, etc.). Then these factors are evaluated in small-time increments (e.g., during 1 day) and the rating values (e.g., Nominal rating of RELY (Reliability) is 1.0 [2]) are entered into the integrated estimation models as shown in Fig. 1. The integrated estimation models will then estimate the effort, schedule, and defect density during the small-time increments. The estimated values represent the dynamic project performance which will provide project managers with a framework for performing trade-off analysis. If the project environment or management decision changes, it will adjust the project driving factors dynamically. The changed project driving factors will be re-evaluated and affect the project performance.

The following sections briefly describe how to combine COCOMO II with system dynamics. The detailed description on this approach can be found in [23].

4.2.1. Bring dynamics to COCOMO II by decomposition

Fig. 2 illustrates how to give the dynamic simulation capabilities to COCOMO II. We decompose the time-aggregated effort and schedule estimates into the time-continuously varying development rate of each phase by using the effort and schedule distribution data provided in [2]. The effort and schedule estimates are time-aggregated in that they just show the end state of the project. The time-aggregated values are decomposed into a small-time incremented development rate (e.g., KsLOC or Function Point (FP)/day) to dynamically simulate the development process. The development rate will be dynamically adjusted during the course of a project in order to reflect the effects of the dynamic project environment.

The expressions in (2) explains the decomposition process.

\[
\text{AverageFP} = \text{PM} / \text{TDEV},
\]

\[
\text{Manpower Rate} = \text{AverageFP} \times \text{Effort%}/\text{Schedule%},
\]

\[
\text{Productivity} = \text{Size}/(\text{PM} \times \text{Effort%}),
\]

\[
\text{Development Rate} = \text{Manpower Rate} \times \text{Productivity}
\]

For example, when the AverageFP is 84 PM/Month and the percentage of the effort and schedule for the requirements phase is 7% and 24% each, the Manpower Rate in requirements phase is 24.5 PM/Month. The percentage of effort and schedule for requirements phase will increase for a large project. We apply the effort and schedule distribution provided by [2] in our simulation tool. The percentage of effort and schedule for each phase is changing depending on the job size and the scale exponent, E. A large project with high exponent has the most effort in test phase and takes the longest time in requirements phase.
The productivity of each phase is the amount of the job size (KSLOC or FP) that can be processed by the unit effort (e.g., one PM or one Person–days) in that phase. The development rate of each phase, product of the manpower rate and productivity, is the amount of work processed in the unit time in each phase. The development rate is applied to each phase through the development life-cycle. This provides the basic structure for the dynamic software process simulation.

4.2.2. Integrate estimation models for trade-off analysis

Fig. 3 shows how several estimation models of COCOMO II are integrated to dynamically analyze the trade-offs among the development effort, schedule, and quality (defect density). It is difficult to increase the quality without increasing either the effort or schedule or both, because the effort, schedule, and quality are highly correlated.

The project driving factors play a pivotal role in integrating estimation models of COCOMO II suite and in making causal relationships among them. As we have discussed in Fig. 1, the rating values of project driving factors evaluated during small-time increments are dynamically changing. Then the evaluated values make the models re-estimate the effort, schedule, and defect density. This provides dynamic trade-off analysis capability.

The “Effort Estimation Model” and “Schedule Estimation Model” in Fig. 3 output the development effort and schedule by using all the project driving factors. The “Defect Introduction Model” uses all the project driving factors except FLEX (Flexibility) and outputs the number of defects introduced. The “Defect Removal Model” outputs the defect density based on the PMAT (Process Maturity) and RELY (Reliability) factors [6]. For example, if the level of development personnel is lowered, the introduced defects are increased and the defect density will be increased. It also increases effort and delays schedule.

We integrate effort, schedule and quality estimation models of COCOMO II with the same input (project driving factors) and make causal relationships among them. This model integration mechanism provides a valuable dynamic trade-off analysis capability and makes the model integration process seamless.

4.2.3. Add effects of dynamic project environment

We have discussed that the equations of the COCOMO II models can be used in dynamic simulation by decomposing the development time into small-time increments. We then described how to integrate effort, schedule, and quality estimation models using the project driving factors. As the last step, we add additional factors such as stakeholder’s friction and developer’s morale and make them interact with project driving factors to reflect changing project environment.

COCOMO II assumes that the project environment is stable, which means the project is “well managed” and the requirements are not substantially changed after the requirements phase [2]. This makes COCOMO II to have only static analysis capabilities. However, Boehm et al. suggest to adjust the ratings of cost drivers during the process when we need to take into account the impact of the risks such as the bad management, personnel turnover, volatile requirements, and aggressive schedule [2].

We apply the system dynamics modeling technique when we analyze the risks such as large amount of creeping requirements, personnel turnover, aggressive schedule, etc. The additional project factors for reflecting the effects of the dynamic project environment or management decision are introduced and related with the project driving factors. Then numerical values are assigned to those relationships for dynamic project performance estimation.

The dynamic project performance estimation capability of our approach will help project managers make better decisions that impact the overall success of the software development project faced with continuously varying project environment.

5. Simulation experiment: a military application development project

We have discussed how to extend the static estimation capability of COCOMO II to dynamic project performance estimation theoretically. This section will describe a simulation experiment on a virtual military application development project. We choose a military domain because military application development projects are quite dynamic in terms of creeping requirements, management changes, etc. Our experience tells us that government managers often neglect the impact of uncontrolled management decision changes.

We configure our simulation tool to represent the typical characteristics of a military project and estimate project performance changes on uncontrolled creeping requirements. We set the simulation tool’s adjustment parameters using an expert judgment technique to represent the dynamic effects caused by uncontrolled creeping requirements which are not considered in COCOMO II.

5.1. Model boundary and assumptions

Large software projects, especially military projects, have a bad reputation for schedule and cost overruns [24]. One of the problems those large projects encountered is changing requirements after requirements phase, which is called creeping requirements. We perform a simulation experiment on a military project which suffers large amount of creeping requirements to show the dynamic effects of it. We have encountered many difficulties in this experiment especially for the lack of project data on the military domain. However, we extract as many data as possible from literature, experts in military domain, and our experience.

We analyze the characteristics of military projects and derive soft factors such as developers’ morale. The soft factors which are psychological behavior of project stakeholders significantly influence the performance of projects. We derive the project factors affected by requirements creep through the analysis on the Jones’s assessment data [25,26], Houston’s study [27], surveys, and interviews.

The soft factors, however, are difficult to quantitatively define. For example, someone may enjoy a challenge and others may not. In this case, our experience in the military domain takes
precedence over the others. As far as we experienced, most of developers who participated in military projects show productivity increase up to a certain point and sharply drop when they have large amount of creeping requirements.

We assume the development life-cycle is the waterfall model and sizing is based on the combinations of Function Points (FP) and Source Lines of Code (SLOC), with counting rules in IFPUG [28] for FP.

Table 1 shows the ratings of COCOMO II drivers for the virtual military software development project. The ratings are based on literature [25,26] and authors’ experience on working many military projects as a project manager. The rationale for the ratings is summarized as follows:

- Product is not similar to several previously developed products: Precedentedness (PREC) is low.
- Development flexibility (FLEX) is low because of various military standards and oversight requirements.
- CMM level 3 is a requirement for receiving government contracts: Process Maturity (PMAT) is high.
- Reliability (RELY) of the software is very high: mission-critical software.
- Product complexity (CPLX) is very high: real-time embedded software.
- Documentation (DOCU) requirements are high due to the elaborate oversight and status reporting criteria.

5.2. Cause–effect analysis on the effects of creeping requirements

Fig. 4 shows the cause–effect diagram which analyzes the qualitative effects of creeping requirements. The arrows in Fig. 4 indicate the direction and polarity of causal influences. For example, the arrow with the positive sign from Creeping Requirements to Job Size means that the increase of creeping requirements also increases job size. On the other hand, the arrow with the negative sign between Friction among Stakeholders and Morale means that the increase of Friction among Stakeholders decreases Morale.

Fig. 4 explains that the technical change, organizational change, and increasing understanding on the developed system increase the end user's additional requirements especially when the project is large-size and long-term. This makes the management or system acquisition organization (acquirer) issues more Request For Change (RFC) to the developer to implement the end user's additional requirements.

At the developer’s side, creeping requirements increase the job size and friction among stakeholders, which causes the conflict among designers, programmers, managers, users, etc. The increased friction among stakeholders decreases the developer’s morale [29], which is a major factor in the staff’s attrition (turnover) and staff’s capability drop [27]. We adjust the ratings of personnel turnover and personnel capability to take into account the effects of the developer’s decreasing morale.

The personnel turnover can be computed by adjusting the rating level of personnel continuity (PCON) and the personnel capability drop can be represented by adjusting the level of ACAP (Analyst Capability) and PCAP (Programmer Capability). We also adjust the relative experience factor ratings (i.e., APEX (Application Experience), PLEX (Platform Experience), and LTEX (Language and Tool Experience)) to assess the effort and duration impacts caused by replacing experienced with less experienced personnel [2]. The dynamically changing driving factors consequently affect the effort, schedule, and quality of the project.

On the other hand, we need to introduce new project factors affected by the increased job size, because the cost drivers cannot accommodate the effects of volatile requirements, which is against the assumption of COCOMO II. We conducted surveys and interviews to elicit the experts’ experience and estimation on project’s performance change when the job size is increasing. Some of survey questions will be discussed in Section 5.3.

The derived project factors which affect the behavioral patterns of effort increase, schedule delay, and quality deterioration are “work intensity”, “manpower delay”, and “quality obstruction” as shown in Fig. 4. The work intensity represents the level of worker's concentration. When the project suffers creeping requirements, managers put pressure on developers to work hard and overwork, which increases the productivity of the developers up to a certain point. It is usually increasing before the developers get tired and sharply decreasing after that. The manpower delay is the delay of manpower commitment for the increasing work. The quality obstruction is the level of hindrance to keep the quality goal. Since the developer’s willingness to keep the quality goal is lowered, it reduces the time and quality for quality assurance activities such as the inspection, review, and test. The rework caused by creeping

Table 1
Ratings of COCOMO II drivers for a military software development project

<table>
<thead>
<tr>
<th>COCOMO drivers</th>
<th>Rate</th>
<th>COCOMO drivers</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREC</td>
<td>L</td>
<td>PRELY</td>
<td>VH</td>
</tr>
<tr>
<td>FLEX</td>
<td>L</td>
<td>DATA</td>
<td>N</td>
</tr>
<tr>
<td>RESL</td>
<td>L</td>
<td>CPLX</td>
<td>VH</td>
</tr>
<tr>
<td>TEAM</td>
<td>N</td>
<td>RUSE</td>
<td>VH</td>
</tr>
<tr>
<td>PMAT</td>
<td>H</td>
<td>DOCU</td>
<td>H</td>
</tr>
<tr>
<td>TIME</td>
<td>N</td>
<td>ACAP</td>
<td>N</td>
</tr>
<tr>
<td>STOR</td>
<td>N</td>
<td>PCAP</td>
<td>N</td>
</tr>
<tr>
<td>PVOL</td>
<td>N</td>
<td>PCON</td>
<td>N</td>
</tr>
<tr>
<td>TOOL</td>
<td>N</td>
<td>APEX</td>
<td>H</td>
</tr>
<tr>
<td>SITE</td>
<td>H</td>
<td>PLEX</td>
<td>H</td>
</tr>
<tr>
<td>SCED</td>
<td>N</td>
<td>LTEX</td>
<td>H</td>
</tr>
</tbody>
</table>

* VL, very low; L, low; N, nominal; H, high; VH, very high; XH, extra high.

Fig. 4. Cause–effect analysis on the effects of creeping requirements.
requirements also increases the defect density due to the increasing complexity of the project.

We have qualitatively analyzed the effects of creeping requirements and derived the most important project factors which are often neglected because they are difficult to quantitatively represent, although they affect the project significantly. To quantitatively evaluate the effects of creeping requirements, we need to assign numerical values to the project factors and their relationships, which will be discussed in Section 5.3.

5.3. Assign numerical values to simulation parameters via surveys

We assign numerical values to those factors by eliciting the experts’ experience using the Delphi technique [3]. We provided the descriptions on the project summarized in Tables 1 and 2 to those who have average 8.7 years experience on military projects. We asked to four experts independently to get their estimates associated with project factors defined in Section 5.2.

The initial job size shown in Table 2 is the work size determined at the end of requirements phase. We asked to estimate the project performance when the average volume of creeping requirements is increasing 10% each from 0% to 100% of the initial job size. We assume the requirements change traffic is 2% per month and increasing pattern of creeping requirements is as shown in Table 2, which is referenced from [25,26].

5.3.1. Effects of increasing friction among stakeholders

Fig. 5 shows the experts’ estimation on the effects of creeping requirements on the friction among stakeholders and developer’s morale. It shows that the level of the friction increases and the developer’s morale decreases when creeping requirements increases from 0.0 to 1.0 (100% increase compared to the initial requirements). We use the ordinal scale to represent the level of stakeholder’s friction to elicit the experts’ estimation as follows:

- 0.0: No conflict
- 0.2: Slight: frequent arguments, disagreements with other stakeholder’s opinion
- 0.4: Some: unwilling to accommodate other stakeholder’s objectives
- 0.6: High: considerably unwilling to accommodate other stakeholder’s objectives
- 0.8: Very High: critical to effort and schedule
- 1.0: Legal Litigation

The following is the scale to represent the level of the developer’s morale:

- 0.0: Lowest: open rebellion
- 0.2: Very Low: very unhappy team
- 0.4: Low: project is a grind
- 0.6: Some dissatisfaction
- 0.8: Slightly down: feel tired a little bit
- 1.0: Normal: happy team

It is difficult to represent these factors numerically, because they represent the emotional belief of human beings. The friction’s effect on morale is especially difficult and dependent on each organization. The decreasing trend of morale in Fig. 5 shows slightly different pattern from the exponential smoothing or adaptive expectation [10] in the sociopsychological theory. Based on the theory, the morale decreases slowly at first, then exponentially decreases. We, however, apply the results as it is in this study to reflect the military domain experts’ experience, perception, and estimation.

Table 3 describes the morale’s effect on turnover rate (PCON) and personnel capability (ACP, PCAP). The average turnover rate is 12% per year based on the simulation configuration in Table 1. We investigate the change of turnover rate when the level of developer’s morale is down. This turnover rate is converted into “Impact_on_PCON_Level” to adjust the rating level of PCON (Nominal 1.0: 12%/year, Low 1.12: 24%/year [2]). The adjusted PCON also adjusts the personnel experience ratings, APEX (Application Experience), PLEX (Platform Experience), and LTEX (Language and Tool Experience) as described in Fig. 4.

The rating level for personnel capability is expressed in terms of percentiles with respect to the overall population of software developers [2]. The organization’s average percentile of the personnel’s capability is 55% [2], which is a Nominal rating, when the morale is in normal state. The ability to analyze, design, and code, efficiency and thoroughness, and the ability to communicate and cooperate are evaluated to be decreased when the developer is getting depressed as shown in Table 3.

Eqs. (3) and (4) show the adjustment of the level of PCON, ACP, and PCAP based on the change of the morale. For example, when morale is dropped to 0.6 (some dissatisfaction state), the turnover rate is increased to 14%/year. Then the level of PCON is adjusted by initial PCON minus “Impact_on_PCON_Level”. We use the enumerated values (VL: 1, L: 2, N: 3, H: 4, VH: 5, XH: 6) for the rating level of cost drivers in the simulation model to interpolate the effort multiplier. In this example, the level of PCON will be 2.83 (3.0–0.17) and hence the effort multiplier of PCON will be 1.02. The increased effort multiplier will increase the development effort.

\[
P_{CON} = Initial_{PCON} - Impact_{on_PCON Level} \quad (A)
\]

\[
(A)PCAP_{Level} = Initial_{Level} - Impact_{on( A)PCAP_{Level}} \quad (4)
\]

5.3.2. Effects of increasing job size

We have derived three project factors, “work intensity”, “manpower delay”, and “quality obstruction”, which affect the behavioral patterns of effort increase, schedule delay, and quality deterioration. These factors are to accommodate the effects of additionally increasing job size, which is against the assumption of COCOMO II. COCOMO estimates the necessary effort and schedule to complete the project with the given fixed job size and required reliability goal. Industry, however, cannot meet the required reliability (quality) goal due to resource shortage and many other related problems when large amount of creeping requirements occur.

Therefore, we need a model adjustment procedure to reflect the behavioral characteristics on large amount of creeping requirements. First, we adjust the rating level of RELY (Reliability) dynamically to reflect the trend of an organization’s quality drop along with the increasing job size, which enables COCOMO to estimate the effort with the adjusted RELY. Next thing we do is applying the trend of an organization’s productivity change and manpower delay along with the increasing job size. This adjustment procedure can be repeatedly applied to precisely customize to a particular organization and is summarized as follows:

<table>
<thead>
<tr>
<th>Life-cycle model</th>
<th>Waterfall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial job size</td>
<td>512 KSLOC</td>
</tr>
<tr>
<td>Creeping requirements</td>
<td>10% increase (0–100%) compared to initial job size</td>
</tr>
<tr>
<td>Change traffic</td>
<td>2% per month</td>
</tr>
<tr>
<td>Pattern of creeping requirements</td>
<td>Design (20%), detailed design (30%), code (30%), test (20%)</td>
</tr>
</tbody>
</table>
Eq. (5) shows this procedure.

\[ \text{QualityObstruction} = \sum \text{RELY Adjustment Up To Expected Quality} \]

iii. Investigate manpower delay: It increases along with the increasing creeping requirements. It adjusts the schedule by controlling the manpower rate.

We assign numerical values to the parameters following the model adjustment procedure. First, we investigate the experts’ estimation on the trend of the development organization’s quality drop (increasing defect density) when creeping requirements are increasing 10% each from 0% to 50% compared to the initial job size. The expected quality of the product in each level of creeping requirements is calculated and then the quality obstruction as the required quality drop to meet the expected quality is computed. The quality obstruction accumulates the adjustment values of decreasing level of RELY factor when the work size is increasing. Eq. (5) shows this procedure.

\[ \text{ExpectedQuality} = \text{InitialQuality} \times \text{QualityDrop} \]

\[ \text{QualityObstruction} = \sum \text{RELY Adjustment Up To Expected Quality} \]

Finally, we investigate the experts’ estimation on the trend of the project’s manpower delay. Table 4 explains that the manpower commitment is delayed 7% compared to the required manpower rate to process the unanticipated increased work when creeping requirements are increased 10% and is delayed 21% when creeping requirements are increased 20%. Eq. (7) explains the adjustment of schedule through the manpower delay.

\[ \text{Manpower Rate Adj} = \text{Normal Rate} \times \text{Manpower Delay} \]

\[ \text{Adjusted Manpower Rate} = \text{Normal Rate} - \text{Manpower Rate Adj} \]

5.4. Model verification and validation

Model verification is to ensure that it is built correctly and model validation focuses on the consistency of the model’s structure and behavior by comparing it with the real world. Greenberger et al. stated that “No model has ever been or ever will be thoroughly validated.” and “useful”, “illuminating” or “inspiring confidence” are more apt descriptors applying to models than “valid” [10,30]. This statement implies that it is difficult to fully validate the model, because historical project data are not enough and project environment varies from organizations to organizations. Given this, the verification and validation are focused on building confidence on the model as a reasonable representation of the system and in its usefulness in the provision of results.

5.4.1. Verification

Our simulation model is distinctive in that the dynamic behavior of the model is derived from the industrially calibrated drivers of COCOMO II and their causal relationships. Therefore, we can verify our model by showing that we integrated several estimation

| Table 4 |
| Trend of the development organization’s manpower delay |
| Requirements creep (%) | 0 | 10 | 20 | 30 | 40 | 50 |
| Manpower delay | 0.0 | 0.07 | 0.21 | 0.23 | 0.28 | 0.43 |
models of COCOMO II correctly. Table 5 shows that our simulation model gives almost the same estimation as COCOMO II. We, then, can reason inductively that if the model is verified, it can represent the general behavioral characteristics of various software projects.

The “Base Case” in Table 5 runs the simulation without any creeping requirements, which is the same condition with COCOMO II. The USC COCOMO II.2000 software [2] implements the development effort and schedule estimation equations of COCOMO II. The defect density for COCOMO II is calculated using the estimation equations of COQUALMO to compare it with our simulation model.

The total effort represents the estimated effort in Person–months (PM) to complete the project from requirements to test phase with the given project configuration shown in Tables 1 and 2. The effort and schedule include the requirements phase, which can be calculated by using the effort and schedule distribution data (e.g., effort and schedule percentage for requirements: 7%, 24% [2]).

This simulation result shows that the simulation model is developed correctly (verified) and can estimate the effort, schedule, and defect density in static environment. Accordingly, the model can represent the normative behavioral characteristics of various types of software development projects by configuring the driving factors of our simulation model.

5.4.2. Validation

Table 6 and Fig. 7 are the simulation results showing the performance change of the military software development project when creeping requirements are increasing. It displays the (relative) increase of effort, schedule, and defect density compared to “Base Case”. For example, when creeping requirements is increased to 50% (average volume of creeping requirements in military domain is 48% of the initial job size [25,26]), the effort increases 44.9% and schedule increases 44.2%. This is a serious project risk factor which forces a project to fail.

Fig. 8 shows the snapshot of the developed tool which is available at [31]. We can change and turn on and off the simulation parameters using the graphical input devices. We use the backfiring tables provided by Jones [25] to convert FP into the equivalent SLOC. The default ratio is 60 SLOC/FP and the users can adjust this by using “Language Converter” knob. Our tool displays the effort for each phase, which is useful to manage human resources when the project plan is away from its normal progress. It can help project managers understand how the project performance changes according to the uncontrolled creeping requirements.

We consulted with the experts about our model’s reasonableness. They agreed that the model’s dynamic project performance estimation on the target domain is very similar with their estimation and our tool’s configuration function, adjustment mechanism, and dynamic performance display are very useful in project management.

Although we cannot show the behavioral consistency of the model with the real world due to the lack of project data, we come up with an agreement with military domain experts that our tool reasonably reproduces the behavioral characteristics of a project suffering large amount of creeping requirements.

5.4.3. Threats to validity

In this section we discuss some of the potential threats to the validity of our approach. We have two potential threats. One is the foundation of using the drivers of COCOMO II in a dynamic simulation. It can be argued that the stretching the COCOMO II model is way beyond its original intentions. In this study, our greatest concern is providing a dynamic estimation capability to those who have not enough historical project data. COCOMO II is an attractive choice in that it already constructed a big database on completed projects. To minimize the potential problem, we utilize the cost drivers which are the most likely to be dynamically adjusted. Adjusting the cost drivers are mentioned and exemplified by the authors of COCOMO II.

The other threat to validity is the limitation of a simulation experiment. It is hard to tell the reasonableness of a simulation model by performing only a single experiment. Although the simulation experiment is performed once, the purpose of the model is well demonstrated. We can estimate the initial effort, schedule, and defect density by using the cost drivers of COCOMO II and then evaluate the impact of dynamic project environment by applying experts’ experience to the parameters of our model.

6. Conclusion and future work

We applied three techniques to dynamically estimate the project performance changes when the project environment is changing. We combined the static estimation models of COCOMO II with system dynamics, and applied expert judgment technique, Delphi, to overcome the limitations of project data.

Table 5

<table>
<thead>
<tr>
<th></th>
<th>Base Case</th>
<th>COCOMO II (include requirements)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total effort (PM)</td>
<td>4995.2</td>
<td>4994.8</td>
</tr>
<tr>
<td>Schedule (months)</td>
<td>69.4</td>
<td>69.455</td>
</tr>
<tr>
<td>Defect density</td>
<td>1.48</td>
<td>1.48</td>
</tr>
</tbody>
</table>

Table 6

<table>
<thead>
<tr>
<th></th>
<th>Base Case</th>
<th>Req. creep (10%)</th>
<th>Req. creep (20%)</th>
<th>Req. creep (30%)</th>
<th>Req. creep (40%)</th>
<th>Req. creep (50%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total effort (PM)</td>
<td>4995.2</td>
<td>5207.8 (1.043)</td>
<td>5497.2 (1.100)</td>
<td>5938.4 (1.189)</td>
<td>6409.7 (1.283)</td>
<td>7239.1 (1.449)</td>
</tr>
<tr>
<td>Schedule (months)</td>
<td>69.4</td>
<td>70.7 (1.019)</td>
<td>73.5 (1.059)</td>
<td>79.5 (1.146)</td>
<td>87.8 (1.265)</td>
<td>100.1 (1.442)</td>
</tr>
<tr>
<td>Defect density</td>
<td>1.48</td>
<td>1.59 (1.074)</td>
<td>1.69 (1.142)</td>
<td>1.86 (1.257)</td>
<td>2.14 (1.446)</td>
<td>2.35 (1.588)</td>
</tr>
</tbody>
</table>

Table 6 and Fig. 7 are the simulation results showing the performance change of the military software development project when creeping requirements are increasing. It displays the (relative) increase of effort, schedule, and defect density compared to “Base Case”. For example, when creeping requirements is increased to 50% (average volume of creeping requirements in military domain is 48% of the initial job size [25,26]), the effort increases 44.9% and schedule increases 44.2%. This is a serious project risk factor which forces a project to fail.

Fig. 8 shows the snapshot of the developed tool which is available at [31]. We can change and turn on and off the simulation parameters using the graphical input devices. We use the backfiring tables provided by Jones [25] to convert FP into the equivalent SLOC. The default ratio is 60 SLOC/FP and the users can adjust this by using “Language Converter” knob. Our tool displays the effort for each phase, which is useful to manage human resources when the project plan is away from its normal progress. It can help project managers understand how the project performance changes according to the uncontrolled creeping requirements.

We consulted with the experts about our model’s reasonableness. They agreed that the model’s dynamic project performance estimation on the target domain is very similar with their estimation and our tool’s configuration function, adjustment mechanism, and dynamic performance display are very useful in project management.

Although we cannot show the behavioral consistency of the model with the real world due to the lack of project data, we come up with an agreement with military domain experts that our tool reasonably reproduces the behavioral characteristics of a project suffering large amount of creeping requirements.

5.4.3. Threats to validity

In this section we discuss some of the potential threats to the validity of our approach. We have two potential threats. One is the foundation of using the drivers of COCOMO II in a dynamic simulation. It can be argued that the stretching the COCOMO II model is way beyond its original intentions. In this study, our greatest concern is providing a dynamic estimation capability to those who have not enough historical project data. COCOMO II is an attractive choice in that it already constructed a big database on completed projects. To minimize the potential problem, we utilize the cost drivers which are the most likely to be dynamically adjusted. Adjusting the cost drivers are mentioned and exemplified by the authors of COCOMO II.

The other threat to validity is the limitation of a simulation experiment. It is hard to tell the reasonableness of a simulation model by performing only a single experiment. Although the simulation experiment is performed once, the purpose of the model is well demonstrated. We can estimate the initial effort, schedule, and defect density by using the cost drivers of COCOMO II and then evaluate the impact of dynamic project environment by applying experts’ experience to the parameters of our model.

6. Conclusion and future work

We applied three techniques to dynamically estimate the project performance changes when the project environment is changing. We combined the static estimation models of COCOMO II with system dynamics, and applied expert judgment technique, Delphi, to overcome the limitations of project data.
The static estimation capability was extended to estimate the dynamic project performance. We integrated the effort, schedule, and quality (COQUALMO) estimation models to analyze the trade-off relationships among them. The additional project factors to represent the effects of the dynamically changing project environment were introduced in our model and the numerical relationships among them were derived from experts’ experience.

Then we implemented a dynamic project performance estimation tool using iThink software and performed a simulation experiment on a virtual military application development project. Certainly, our experiment is performed on non-real project, but we configured our tool to represent the general characteristics of a military domain project and ran a simulation experiment which suffered unanticipated and uncontrolled requirements creep.

The simulation results showed that our approach can estimate the effort, schedule, and defect density changes dynamically and visualize the behavioral trend of a project on creeping requirements. This helps project managers understand how the project performance changes according to the uncontrolled creeping requirements.

Our tool was verified by showing that the integrated estimation models in our dynamic simulation model estimated equivalent values compared to the estimation models of COCOMO II. We also tried to validate our approach by showing that the behavioral characteristics of a project performance change of our model was comparable with that of the experts’ estimation. However, it was extremely difficult to accurately estimate the project performance in dynamic project environment. It is because each organization shows different patterns and reasons of quality deterioration, effort increase, and schedule delay. Furthermore, it is hard to quantitatively represent human’s perceptions.

On the other hand, we understand that our method is not complete, but will give at least a rough guideline for project managers.

Our approach is unique in that it explicitly represents the changes of effort, schedule, and defect density of each development phase. It also describes the internal mechanism of the model and provides an executable simulation model for others.

As future work, it will be necessary to analyze the sensitivity of input parameters and collect relevant project data to improve the estimation accuracy. It is also interesting to make an experiment on the effects of management’s project control. The project manager can control the issuing rate of RFC (Request For Change) based on requirements management policies and also can control the project deadline and quality goal to meet the project objectives. It will be helpful if we can predict the effects of management’s activities before actually implementing them.

![Fig. 7. Behavioral trend of a military project with creeping requirements.](image-url)
Acknowledgements

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Advancement) (IITA-2007-(C1090-0701-0032)). This work was partially supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract.

References


Fig. 8. Snapshot of the dynamic project performance estimation tool.