Project Delay Variability Simulation in Software Product Line Development

Makoto Nonaka[1], Liming Zhu[2], Muhammad Ali Babar[3], and Mark Staples[2]

[1] Faculty of Business Administration, Toyo University, Japan
nonaka-m@toyonet.toyo.ac.jp
[2] National ICT Australia
{liming.zhu,mark.staples}@nicta.com.au
[3] Lero, University of Limerick, Ireland
Muhammad.alibabar@ul.ie

Abstract. The possible variability of project delay is useful information to understand and mitigate the project delay risk. However, it is not sufficiently considered in the literature concerning effort estimation and simulation in software product line development. In this paper, we propose a project delay simulation model by introducing a random variable to represent the variability of adaptive rework. The model has been validated through stochastic simulations by comparing generated adaptive rework to an actual change effort distribution, and by sensitivity analysis. The result shows that the proposed model is capable of producing reasonable variability of adaptive rework, and consequently, variability of project delay. Analysis of our model indicates that the strength of dependency has a larger impact than the number of residual defects, for the studied simulation settings. However, high levels of adaptive rework variability did not have great impact on overall project delay.

Key words: process simulation, software product line development, product quality, project planning

1 Introduction

Software Product Line (SPL) development can shorten the total cycle time, the duration from the beginning of core asset development to the end of product development, by achieving large-scale reuse [1]. However, effort estimation, planning, and development management for SPL are more complex and difficult than those for sequential development, because of inter-connected relationships between core assets and products, concurrency of their projects, and multiple deadline management [2]. In addition, there are still general problems with software effort estimation errors such as unplanned work [3] as well as requirements volatility [4]. The total cycle time can sometimes be longer than initially planned because of these problems.

One source of unplanned work is poor quality of software artifacts. A certain number of defects will inevitably remain in released core assets, as software testing can not demonstrate the absence of defects [5]. When residual defects in core

assets are detected after their release to product projects (not to customers), corrective maintenance[4] is usually performed[5] to modify the core assets. When multiple product projects are undertaken simultaneously during core asset maintenance phase, corrective maintenance in core assets sometimes brings associated rework to all ongoing product projects that depend on the core assets, to adapt the products to the changed core assets. We call this type of rework "adaptive rework".[6]

With regard to this problem, we previously proposed a simulation model for estimating project delay in concurrent software development and conducted a deterministic simulation with fictional project data [7], which did not estimate the variability of project delay. The variability, or the level of risk of project delay is useful information [8] when a project manager wants to understand and mitigate project delay risk. Even in the literature concerning effort estimation and simulation, the level of risk of project delay in SPL development has not been considered enough [9–12]. In consideration of the variability of project delay, we set the following research questions in this paper. *How much variability of project delay in SPL development is expected when (a) the number of residual defects in core assets changes and (b) the strength of dependency changes?*

To explore these research questions, we propose a simulation model for estimating project delay and its variability by introducing a random variable to represent the duration of adaptive rework. Furthermore, we increase the expressiveness of the model by introducing inter-dependency of core assets. We conducted stochastic simulations with fictional project data with the proposed model.

The reminder of this paper is organized as follows. Sect. 2 describes the proposed simulation model which includes the previous model and the enhanced features. Simulation results and derived implications are described in Sect. 3. Sect. 4 discusses model evaluation. Sect. 5 contains a discussion and describes related work. Concluding remarks are described in Sect. 6.

2 Proposed Simulation Model

Software process analysis approaches can be categorized into the following three types [13]: analytical models such as COCOMO II [9], continuous simulation models [14, 15], and discrete-event simulation models [16–18]. A discrete-event simulation model is suitable for detailed analyses of process and project performance [19]. As we consider sequential events concerning residual defects and

[4] Corrective maintenance is defined in an IEEE standard [6] as "reactive modification of a software product performed after delivery to correct discovered faults."

[5] In actual practice, not all discovered defects will always be fixed.

[6] The meaning of 'adaptive rework' in this paper and that of 'adaptive maintenance' in an IEEE standard [6] are somewhat different. Adaptive maintenance is defined in [6] as "modification of a software product performed after delivery to keep a computer program usable in a changed or changing environment."

adaptive rework, we apply a discrete-event simulation model to the proposed model.

2.1 Primary Factors of the Simulation Model

Suppose that there is a limitation on available resources. To avoid or reduce project delay, the frequency of adaptive rework as well as its duration should be reduced. The frequency is closely correlated with the number of residual defects in core assets. The duration of each piece of adaptive rework will in practice relate to the strength of dependency between core assets and products. This assumption is partly supported by [20–22] showing that design complexity has a large influence on maintenance effort. The duration will also relate to what development phase it occurs in. Literature reports that the ratio of the cost of finding and fixing a defect during design, test, and field use is 1 to 13 to 92 [23] or 1 to 20 to 82 [24].

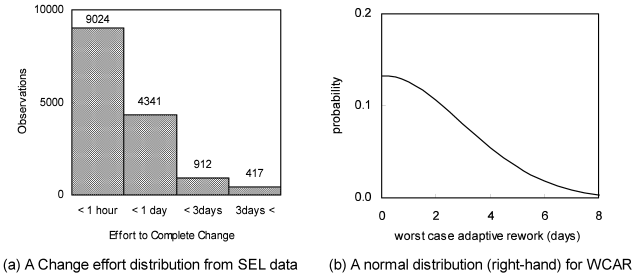
From this discussion, we select the following three factors as primary factors of the simulation model.

1. *The number of residual defects in core assets (NRD).* NRD will depend on product size, product complexity, process quality, and other factors. We assume that NRD can be estimated.
2. *The strength of the dependency (DEP).* We consider DEP between core assets and products as well as among core assets. DEP is represented as a continuous variable that ranges from 0 to 1. DEP = 0 means no dependency, and DEP = 1 means the strongest. In practice, there may be different levels of dependency for different changes, but as discussed below, we use a single DEP value to represent the worst-case dependency.
3. *Work effort multiplier (WEM).* We introduce WEM to represent the ratio of the duration of pieces of adaptive rework for each development phase in which adaptive rework occurs. We assume that each product project follows sequential processes. WEM is represented as a continuous variable that ranges from 0 to 1.

2.2 Determining Adaptive Rework

To determine the duration of each piece of adaptive rework, we first consider defect correction completion time in the core asset maintenance phase that determines the time when adaptive rework occurs. The defect correction completion time can be determined by applying a Software Reliability Growth Model (SRGM) [25]. Suppose that all residual defects in core assets are detected during core asset maintenance phase. If we draw an SRGM curve during the phase, the defect correction completion time of these defects can be determined by assigning a time to each defect along with the curve depending on reliability growth.

Next, we introduce a parameter "worst case adaptive rework" (WCAR). WCAR is supposed to represent the duration of adaptive rework in the following worst-case scenario: (1) the defect correction completion time is at the end of the product project, and (2) DEP is the strongest.



(a) A Change effort distribution from SEL data (b) A normal distribution (right-hand) for WCAR

Fig. 1. An actual error correction effort histogram and a distribution for WCAR.

WCAR inherently has a certain distribution, because the duration of WCAR depends on what kind of defects corrected in core assets. Here we introduce a continuous random variable to represent the WCAR distribution. According to the Software Engineering Laboratory (SEL) data subset [26], an effort distribution for error correction has a right-skewed distribution as depicted in Fig. 1 (a). To generate a WCAR distribution like Fig. 1 (a), we use the right-hand half part of a normal distribution (Fig. 1 (b), $\mu = 0$ and $\sigma = 3$, for example). Note that the range of the WCAR distribution is larger than that of the SEL data distribution, as the WCAR distribution represents worst cases of adaptive rework instead of actual change effort.

With these parameters, the duration of each piece of adaptive rework can be determined as follows. The duration of adaptive rework $\Delta r_i(d_j)$ (in months) caused by the defect d_j in the product project i is assumed to be represented by the formula

$$\Delta r_i(d_j) = \mathrm{EffDist}^{-1}_{wcari}(p) \times \mathrm{WEM}_j(t_{d_j}) \times \mathrm{DEP}_{ki} \times \epsilon, \tag{1}$$

where $\mathrm{EffDist}^{-1}_{wcari}(p)$ (in months) is the inverse function of the WCAR effort distribution probability function for the project i. Probability p is given at random. WEM for the project i is represented with $\mathrm{WEM}_j(t_{d_j})$ when the defect d_j correction is completed in core asset maintenance phase at the time t_{d_j}. DEP between the core assets k and the product i (or core assets i) is represented with DEP_{ki}. The parameter ϵ is 1 if t_{d_j} is within the period of the product project i. Otherwise, ϵ is 0.

2.3 Model Assumptions

The simulation model relies on the following assumptions:

1. Adaptive rework occurs at the time when the causal defect is corrected. Actually, this assumption is not true in practice. Defect correction delay has been observed in [27], which reported that 55% of defects were corrected within a few days, 36% within the next week, and the last 9% before customer release or in the next version.

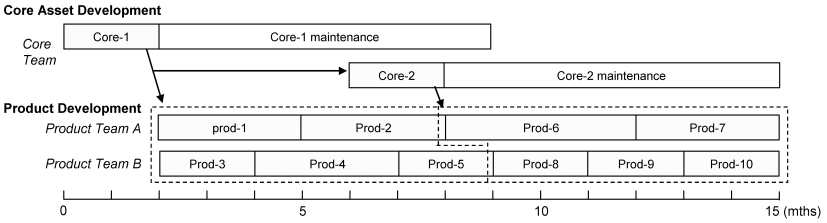


Fig. 2. Time schedule of the fictional SPL development project.

2. The amount of adaptive rework decreases from WCAR, depending on DEP and WEM, which is partly supported by [20–24].
3. Adaptive rework for completed projects is not performed even though later defect corrections in dependent core assets may be performed.
4. Products are sequentially developed in planned order by an assigned team with a limited number of resources.
5. The impact of imperfect defect correction during corrective maintenance in core assets and adaptive rework is negligible, which is in practice supported by [27]. That is, it makes little difference on project delay if we do not consider defect correction effort arisen from the another defects that will be injected during those activities.

3 Simulation Results

3.1 Project Data and Parameters

A fictional SPL development project has been studied for simulation. The time schedule of the project is shown in Fig. 2. Arrows in Fig. 2 represent dependency. In this project, 10 products are scheduled to be developed by two product teams concurrently. Core assets are developed, maintained, and enhanced by a core team that is independent of the product teams. Core-2 is an enhanced version of Core-1. Prod-1 to Prod-5 depend on Core-1, while Prod-6 to Prod-10 depend on Core-2. Core-1 maintenance phase is scheduled to be finished at the same time when Prod-5 finishes. The scheduled total cycle time is 15 months. Each pair of successive product projects is scheduled without any buffers. The duration of core asset maintenance phase will be expanded in response to the delayed product projects.

Note that the absolute sizes of core assets and products are not considered here, because they do not directly affect simulation results in the proposed model. Nonetheless, size does affect NRD as described in Sect. 2.1, and DEP might be partly dependent on size.

Several patterns for NRD and DEP have been studied to explore the research questions. For the other parameters, a fixed value or a fixed model is applied.

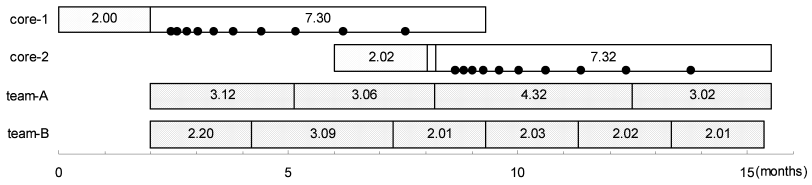


Fig. 3. A simulation result: detail view of project delay.

1. *NRD:* Four patterns of NRD have been studied ranging from 10 to 40 defects in increments of 10 defects. These values are the sum of NRD in both Core-1 and Core-2.
2. *DEP:* We have studied three DEP levels of 0.2, 0.6 and 1.0.
3. *WEM:* By considering the empirical data concerning the cost of defect correction during design and test [23, 24], a factor of 20 has been studied. To make the model simple, we use a linear model ranging from 0.05 to 1.0.
4. *Defect correction completion time:* Though numerous SRGMs have been proposed in the literature [25], we apply the following simple logarithmic function
$$y = 1 + \log_a x, \tag{2}$$
where y represents cumulative rate of defect detection, while x represents normalized duration of core asset maintenance phase ($0 < x \leq 1$). In this simulation $a = 20$ is used, which means that 60% of residual defects are corrected before 30% of maintenance phase, and 90% of defects are corrected before 75% of the phase, for example.
5. *WCAR:* We use the distribution pattern in Fig. 1 (b). Note that WCAR is limited up to 8 days in the simulations in order not to generate unrealistically large amount of rework, though a normal distribution has unlimited values.

3.2 Result 1: Detail View of Project Delay and Adaptive Rework

Figure 3 shows a simulation result representing how project delay occurs caused by residual defects in detail (DEP = 0.6, NRD = 20). The dots represent residual defects and their correction completion time. One can see that Core-2 development project is delayed for 0.02 months due to two residual defects detected in Core-1 maintenance phase. The estimated total cycle time is 15.39 months (i.e. a total delay of 0.39 months).

Figure 4 shows the histograms of generated adaptive rework with four combinations of NRD and DEP. Note that each boxplot has a different scale in both x-axis and y-axis. The shapes of the histograms are all skewed to the right, as the WCAR distribution is also right-skewed. The ranges of Fig. 4 (c, d) are quite smaller than those of Fig. 4 (a, b). The ranges of Fig. 4 (a, b) are still smaller than those of the WCAR distribution in Fig. 1 (b), as the WCAR distribution is assumed to have the largest WEM. In Fig. 4 (c, d), all pieces of adaptive

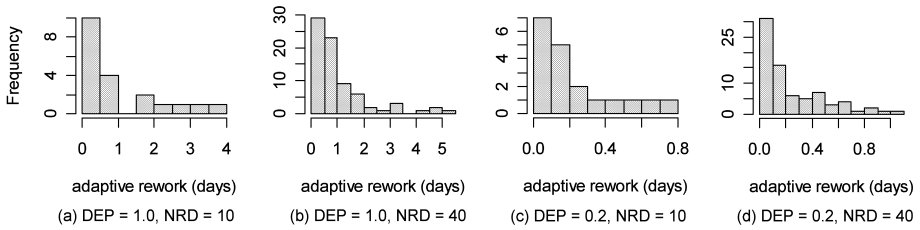
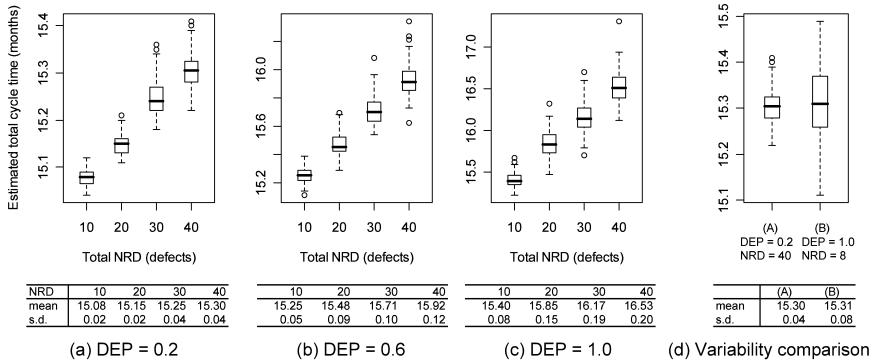


Fig. 4. Examples of generated adaptive rework histograms.



NRD	10	20	30	40
mean	15.08	15.15	15.25	15.30
s.d.	0.02	0.02	0.04	0.04

	10	20	30	40
	15.25	15.48	15.71	15.92
	0.05	0.09	0.10	0.12

	10	20	30	40
	15.40	15.85	16.17	16.53
	0.08	0.15	0.19	0.20

	(A)	(B)
mean	15.30	15.31
s.d.	0.04	0.08

Fig. 5. Simulation results on estimated total cycle time (Note: y-scales are different).

rework are completed within one day and most of them are less than 0.2 day (two or three hours) because of weak DEP. The distributions with weak DEP are considered to be a better approximation of actual change effort distribution shown in Fig. 1 (a).

3.3 Result 2: Variability of Project Delay

We conducted 100-run simulations for each combination of NRD and DEP. The boxplots in Fig. 5 (a, b, c) represent the simulation results. The mean and the standard deviation of each combination are shown in the table below the boxplot. Note that each y-axis has a different scale among boxplots.

The results imply that project delay and its variation can be held down if DEP and NRD are low (DEP = 0.2 and NRD = 10). Even for the worst combination in the studied settings (DEP = 1.0 and NRD = 40), the standard deviation of estimated project delay was not very large (0.20). In this case, the range for all data including suspected outliers was from 16.12 to 17.31. As the initial planned time was 15 months, the estimation error rate ranges from 1.07 to 1.15. It means that 8 percentage points of schedule estimation error has appeared in this case at most. It is considered to be in practice quite a small difference

for effort or schedule estimation error. When we consider the impact of DEP on durations of pieces of adaptive rework, it sometimes bring larger durations (over one or two days) of adaptive rework as shown in Fig. 4 (a, b). However, the overall effect on project delay is trivial according to the simulation result.

The following is a detailed analysis of the simulation results.

1. *Magnitude of variability:* The standard deviations for DEP = 0.2 are quite small (from 0.02 to 0.04), and even those for DEP = 1.0 are still small (from 0.08 to 0.20). This is because most pieces of adaptive rework are distributed among smaller values regardless of DEP.
2. *Difference of variability in NRD:* The standard deviations of the same DEP slightly increase as NRD increases, because the chance to have more pieces of adaptive rework also increases. By comparing the pair of both (a, b) and (c, d) in Fig. 4, one can see that the frequencies of (b) and (d) are larger than those of (a) and (c) respectively, and that a few but large durations of pieces of adaptive rework are appeared in both (b) and (d).
3. *Difference of variability in DEP:* Similarly, the standard deviations of the same NRD increase as DEP increases. DEP has a stronger impact on variability compared to NRD, when we consider only for the studied simulation settings. This is because different DEPs generate different WCAR distributions, while different NRDs share the same WCAR distribution. The shape of a WCAR distribution is considered as a dominant factor on variability, rather than NRD.
4. *Comparison of variability:* We selected two simulation settings which have almost the same estimated total cycle time but different parameters: (A) DEP = 0.2 and NRD = 40, and (B) DEP = 1.0 and NRD = 8. Fig. 5 (d) shows the comparison results between them. To judge whether the means of both settings are the same, we used Welch's t-test at the 5% significance level. The p-value was 0.40, so we can conclude that there is no statistically significant difference on means between them. However, an F-test showed quite a small p-value $\ll 0.01$. Then we can conclude that there is a significant difference between their variances. This difference mostly comes from the different WCAR distributions, as described in the item 3.

4 Model Evaluation

Because of the nature of simulation study, it is impossible to validate all aspects of the proposed simulation model comprehensively. However, the utility of the model can be evaluated by using empirical data, even though it will not demonstrate comprehensive validation. As we do not have enough empirical data at this moment, we follow four aspects of validation and verification for simulation models [28]: conceptual model validity (between problem entity and conceptual model), computerized model verification (between conceptual model and computerized model), operational validity (between problem entity and computerized model), and data validity.

Conceptual model validity and *data validity:* The proposed model is considered to be reasonably valid under the assumptions described in Sect. 2.3, because the formula (1) is partly supported by several empirical observations as stated in Sect. 2.1 and 2.2. Data validity as input to the model is also supported by these empirical observations in terms of determining the WCAR distribution and WEM. However, there are some limitations of the model. We discuss this topic in Sect. 5.2.

Computerized model verification: We have confidence that the simulation program is accurately implemented because of our precise investigations of the simulation results (like Fig. 3 and Fig. 4), and because of our inspections of the individual simulation runs including extreme conditions.

Operational validity: In general, operational validity is difficult to assess when no observable problem entity is available. In such a case, comparison to other models and sensitivity analysis are meaningful approaches to validate a simulation model [28]. One possible approach is to compare the proposed model to other effort estimation models. However, this approach is not applicable in this case, because both COCOMO II [9] and COPLIMO [10], a COCOMO II based cost estimation model for SPL development, do not produce variability of estimated effort. These models have a lot of parameters such as effort multipliers, but these parameters are deterministic but not stochastic.

Another possible approach is to evaluate the generated adaptive rework by the simulation program rather than total cycle time. By comparing the distributions of the generated adaptive rework in Fig. 4 (a, b) to the change effort distribution from the SEL data in Fig. 1 (a), both distributions can be subjectively judged to be similar. However, the ranges of Fig. 4 (c, d) are smaller than that of Fig. 1 (a), as DEP has a strong impact on durations of adaptive rework. At least, we can conclude that the simulation model is capable of producing reasonable adaptive rework distributions.

Sensitivity analysis is also a useful approach to demonstrate validity of the model, which we have already discussed in Sect. 3.3. It can be considered that the model has reasonable validity but some limitations described in Sect. 5.2.

5 Discussion and Related Works

5.1 Calibration for Practical Application

When one wants to apply the proposed model in practical situations, the parameters of the model have to be calibrated. NRD and WEM may be able to be estimated easily by investigating one's own organizational defect correction data. The WCAR distribution model might be generated by measuring adaptive rework caused by residual defects. However, DEP will be difficult to calibrate, though it has a stronger impact on duration of adaptive rework compared to NRD, for the studied settings.

In this paper, specific DEP metrics are not assumed. DEP might depend on attributes such as coupling between core components and product components,

number of dependent product components reusing a core component, and inheritance depth between core and product components. Those attributes and measured values will be translated into DEP and calibrated by checking generated adaptive rework distributions like Fig. 4.

5.2 Limitations of the Model

The proposed model uses the calendar time scale for the duration of adaptive rework instead of the effort or cost scale, because defect correction completion time is also represented by using the calendar time scale. Therefore, a project delay always occurs corresponding to any residual defects, even though the durations of pieces of adaptive rework are very short. In practice, such small pieces of adaptive rework may not bring delay, but instead require additional effort or cost. This is one of the limitations of the model in terms of conceptual validity.

In addition, the project delay estimated by the proposed model can not be translated into absolute effort or cost, as the current model does not use those scales directly. However, when considering relative effort or cost estimation error, the current model may be useful as it is.

Moreover, the current model does not explicitly consider resource limitation and resource allocation policies as well. Project delay will be occurred in practice when enough resource are not available. There are other sources of project delay such as unplanned work arisen from requirements change and defect correction. We are in the process of introducing these factors into the simulation model.

5.3 Effort Estimation and Simulation in SPL Development

Several studies have appeared in the literature on estimating the benefits of SPL development [10, 29, 30]. These studies use more macro-level analytical models compared with our model. The primary purpose of the studies [29, 30] is for estimating the return on investment of SPL development compared with non-SPL development. COPLIMO [10] is a deterministic cost estimation model for SPL and does not represent uncertainty, as well as COCOMO II [9]. COCOMO-U [12] introduces uncertainty into COCOMO II, but does not mention how the model can be applied to SPL development.

Chen et al. proposed a discrete-event SPL process simulator using COPLIMO as their base cost model [11]. Schmid et al. studied SPL planning strategies through deterministic simulations [2]. These two studies have similar research questions to ours. However, these studies do not explicitly use factors such as NRD, DEP, and adaptive rework. They are also not capable of calculating the level of risk of estimated effort under uncertainty, as they are based on deterministic simulation models.

6 Conclusions

In this paper, we proposed a stochastic simulation model for estimating project delay and its variability in SPL development. The model has been validated

through simulations with fictional project data, by comparing generated adaptive rework to an actual change effort distribution, and by sensitivity analysis. The result shows that the proposed model is capable of producing reasonable variability of adaptive rework, and consequently variability of project delay, even though some limitations exist. Analysis of our model indicates that the strength of dependency, or DEP, has a larger impact on durations of adaptive rework than the number of residual defects, or NRD, for the studied simulation settings. The result shows that the level of risk of project delay can be held down if DEP and NRD are quite small. It will still be held down even though DEP is strong, if most pieces of adaptive rework do not require large effort. When we consider the impact of DEP, it sometimes bring larger durations of adaptive rework. However, the overall effect on project delay is trivial according to the simulation result.

The future work primarily involves empirical validation of the proposed model, enhancement of the model to overpass the limitations and the model assumptions which constrain the utility of the model, and calibration methods of the parameters. We are in the process of enhancing the model to be capable of estimating absolute effort overruns under specific resource allocation plan as well as its limitation. We are also trying to contact some companies to gather empirical SPL development data that are usable for model evaluation.

Acknowledgments. This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Young Scientists (B), 16700042, 2005. National ICT Australia is funded through the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council. The third author was working with NICTA when this paper was produced.

References

1. Clements, P., Northrop, L.M.: Software Product Lines: Practices and Patterns. Addison-Wesley, MA (2001)
2. Schmid, K., Biffl, S.: Systematic management of software product lines. Softw. Process Improve. Pract. **10** (2005) 61–76
3. Genuchten, M.v.: Why is software late? an empirical study of reasons for delay in software development. IEEE Trans. Softw. Eng. **17**(6) (1991)
4. Subramanian, G.H., Breslawski, S.: An empirical analysis of software effort estimate alterations. J. Systems and Software **31**(2) (1995) 135–141
5. Dijkstra, E.: Notes on structured programming. In Dahl, O.J., Dijkstra, E., Hoare, C.A.R., eds.: Structured Programming. Academic Press, London (1972)
6. IEEE: Ieee std. 1219-1998, ieee standard for software maintenance (1998)
7. Nonaka, M., Azuma, M.: Software delivery estimation model for incremental and iterative development process considering undetected design defects (in japanese). In: Proc. Software Symposium 2003. (2003) 107–114
8. Jørgensen, M.: Realism in assessment of effort estimation uncertainty: It matters how you ask. IEEE Trans. Softw. Eng. **2004**(30) (2004) 209–217

9. Boehm, B.W., Abts, C., Brown, A.W., Chulani, S., Clark, B.K., Horowitz, E., Madachy, R., Reifer, D., Steece, B.: Software Cost Estimation with COCOMO II. Prentice Hall (2000)
10. Boehm, B.W., Brown, A.W., Madachy, R., Yang, Y.: A software product line life cycle cost estimation model. Proc. 2004 Intl. Symp. Empirical Softw. Eng. (ISESE'04) (2004) 156–164
11. Chen, Y., Gannod, G.C., Collofello, J.S.: A software product line process simulator. Softw. Process Improve. Pract. **11** (2006) 385–409
12. Yang, D., Wan, Y., Tang, Z., Wu, S., He, M., Li, M.: Cocomo-u: An extension of cocomo ii for cost estimation with uncertainty. Lecture Notes in Computer Science **3966** (2006) 132–141
13. Donzelli, P.: A decision support system for software project management. IEEE Software **23**(4) (2006) 67–75
14. Abdel-Hamid, T., Madnick, S.: Software Project Dynamics- An Integrated Approach. Prentice-Hall, Englewood Cliffs, NJ (1991)
15. Calavaro, G.F., Basili, V.R., Iazeolla, G.: Simulation modeling of software development process. In: Proc. 7th European Simulation Symposium. Soc. for Computer Simulation. (1995)
16. Hansen, G.A.: Simulating software development processes. IEEE Computer **29**(1) (1996) 73–77
17. Antoniol, G., Cimitile, A., Lucca, G.A., Penta, M.: Assessing staffing needs for a software maintenance project through queuing simulation. IEEE Trans. Software Eng. **30**(1) (2004) 43–58
18. Padberg, F.: A study on optimal scheduling for software projects. Softw. Process Improve. Pract. **11** (2006) 77–91
19. Kellner, M.I., Madachy, R.J., Raffo, D.M.: Software process simulation modeling: Why? what? how? J. Systems and Software **46**(2-3) (1999) 113–122
20. Epping, A., Lott, C.M.: Does software design complexity affect maintenance effort? In: Proc. 19th Softw. Eng. Workshop. (1994) 297–313
21. Bocco, M.G., Moody, D.L., Piattini, M.: Assessing the capability of internal metrics as early indicators of maintenance effort through experimentation. J. Software Maintenance and Evolution **17**(3) (2005) 225–246
22. Ramanujan, S., Scamell, R.W., Shah, J.R.: An experimental investigation of the impact of individual, program, and organizational characteristics on software maintenance effort. J. Systems and Software **54** (2000) 137–157
23. Kan, S.H., Dull, S.D., Amundson, D.N., Lindner, R.J., Hedger, R.J.: As/400 software quality management. IBM Systems Journal **33**(1) (1994) 62–88
24. Remus, H.: Integrated software validation in the view of inspections / reviews. In: Proc. Symposium on Softw. Validation, Elsevier (1983) 57–64
25. Musa, J.D.: Software Reliability Engineering. Osborne/McGraw-Hill (1998)
26. SEL: Sel (software engineering laboratory) data. http://www.cebase.org (1997)
27. Defamie, M., Jacobs, P., Thollembeck, J.: Software reliability: assumptions, realities and data. In: Proc. 1999 Intl. Conf. Softw. Maintenance (ICSM'99). (1999) 337–345
28. Sargent, R.G.: Validation and verification of simulation models. Proc. 31st Conf. Winter Simulation (1999) 39–48
29. Cohen, S.: Predicting when product line investment pays. Technical Report Techinical Report CMU/SEI-2003-TN-017, Software Engineering Institute, Carnegie Mellon University (2003)
30. Böckle, G., Clements, P., McGregor, J.D., Muthig, D., Schmid, K.: Calculating roi for software product lines. IEEE Software **21**(3) (2004) 32–38