By Hoh Peter In, Jongmoon Baik,
Sangsoo Kim, Ye Yang, and Barry Boehm

A QUALITY-BASED COST ESTIMATION MODEL FOR THE PRODUCT LINE LIFE CYCLE

In reusing common organizational assets, the software product line (SPL) provides substantial business opportunities for reducing the unit cost of similar products, improving productivity, reducing time to market, and promoting customer satisfaction [4]. By adopting effective product line practices, return on investment (ROI) becomes increasingly critical in the decision-making process. The majority of SPL cost estimation and ROI models [5–9] confine themselves to software development costs and savings. However, if software quality cost is considered in the spectrum of the SPL life cycle, product lines can result in considerably larger payoffs, compared to non-product lines.

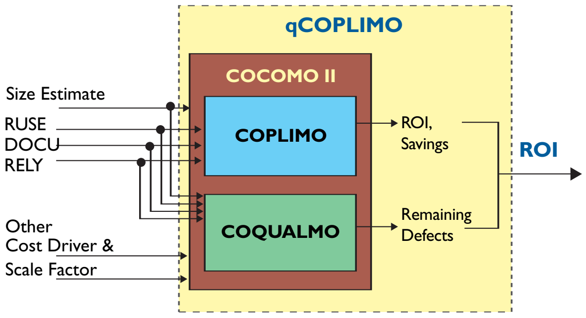

Figure 1. Overview of qCOPLIMO.

This article proposes a quality-based product line life cycle cost estimation model, called qCOPLIMO, and investigates the effect of software quality cost on the ROI of SPL. qCOPLIMO is derived from two COCOMO suite models: COPLIMO and COQUALMO, as presented in Figure 1. COPLIMO [2] provides a baseline cost estimation model of the product line life cycle, and COQUALMO [3] estimates the number of residual defects. These models are used to estimate software quality cost. Both models are an extension of COCOMO II [1].

Quality-Based SPL Cost Estimation Model (qCOPLIMO)

Existing SPL cost estimation models [5–9] do not significantly consider the software quality cost, which is spent on removing undetected defects after product release. In general, the future costs for correction of defects undetected at product release consume a large portion of total maintenance costs. The proposed model is recommended, which includes the software quality cost in the SPL business case analysis. qCOPLIMO consists of the following two cost models: Relative Cost of Writing for Reuse (RCWR) for initial product line development and Relative Cost for Reuse (RCR) for the following product development cases.

Relative Cost of Writing for Reuse (RCWR): RCWR is the added cost of writing software to be most cost-effectively reused across a product line family of applications, relative to the cost of writing a standalone application. The software quality cost is added to the baseline SPL for RCWR ($\text{COPLIMO}_{\text{RCWR}}$), proposed in [2], as follows:

$$\begin{aligned} C_{\text{RCWR}} &= \text{LaborRate} * \text{COPLIMO}_{\text{RCWR}} + \text{Software Quality Cost}_{\text{RCWR}} \\ &= \text{LaborRate} * [\text{COCOMO baseline (initial software size)} * \text{Effort Adjustment for RCWR}] + \\ &\quad [\underline{\text{C}}\text{ost per }\underline{\text{D}}\text{efect} * (1 - \underline{\text{T}}\text{esting }\underline{\text{E}}\text{ffectiveness}) * \text{COQUALMO (initial software size, } \text{EM}_{\text{PL}})], \end{aligned}$$

where EM_{PL} is the $\underline{\text{E}}$ffort $\underline{\text{M}}$ultiplier of the COCOMO II cost drivers for the product line development and COCOMO baseline is calculated as $2.94 * (\text{software size})^{1.0997} * \Pi\,(\text{EM})$

Relative Cost for Reuse (RCR): RCR is the cost of reusing the software in a new application with the same product line family, relative to developing newly built software for the application. After the initial product is developed using product line engineering practice, which concentrates on development for future reuse, the portion or the whole can be used for other products in the same product line family. Like RCWR, the software quality cost is added to the baseline SPL cost for RCR ($\text{COPLIMO}_{\text{RCR}}$), [2], as follows:

$$\begin{aligned} C_{\text{RCR}} &= \text{LaborRate} * \text{COPLIMO}_{\text{RCR}} + \text{Software Quality Cost}_{\text{RCR}} \\ &= \text{LaborRate} * [\text{COCOMO baseline (software size for reuse)}] + \\ &\quad [\underline{\text{C}}\text{ost per }\underline{\text{D}}\text{efect} * (1 - \underline{\text{T}}\text{esting }\underline{\text{E}}\text{ffectiveness}) * \text{COQUALMO (software size for reuse, } \text{EM}_{\text{PL}})] \end{aligned}$$

The estimated quality-based SPL cost for developing N products is as follows:

$$C_{\text{PL}}(\text{N}) = C_{\text{RCWR}} + (N-1) * C_{\text{RCR}}$$

where N is the number of products to be developed in SPL ....(Eq.1)

Parameters	Values	Parameters	Values
Initial Software Size	100 KSLOC	Software size for reuse	50.11 KSLOC
LaborRate	\$8,000 / MM	Effort Adjustment for RCWR[1]	1.469362
EM_{NPL} (all cost drivers are Nominal)	1.0	Testing Effectiveness (TE)	0.9
EM_{PL}[2]	1.78227	Cost per Defect (CD)	\$10,000

[1] Effort Adjustment is calculated by *PFRAC + RCWR * (AFRAC+RFRAC))*. PFRAC (the Product-specific FRACtion of the software product's size) is the portion of the software unique to the particular product of a product family. The Adapted-software FRACtion of the software (AFRAC) is the portion of the product line software that must be modified to operate effectively. The Reused-software FRACtion of the software (RFRAC) is the portion of product line software that can be reused without modification, as a black box. AFRAC and FRRAC are adjusted by the higher ratings (that is, RCWR) of RUSE, RELY, and DOCU. For the detailed information, see [2].

[2] If the product line has considerable variation, its development-for-reuse factor RUSE (reuse) can be rated as Very High, with an effort multiplier of 1.15. In order to minimize software understanding penalty a Very High DOCU (documentation) rating, with an effort multiplier of 1.23, and Very High RELY (reliability) rating, with an effort multiplier 1.26 can be elected. EM_{PL} is then (1.15)•(1.23)•(1.26)= 1.78227.

Primary ROI Input Parameters.

ROI Analysis of a Quality-Based Product Line: A Case Study

The effect of software quality cost on ROI figures for product line vs. standalone product development is investigated using a representative example, with the parameters shown in the table here based on collected 161 real-world industrial COCOMO II data, and experience in aircraft and spacecraft product line domains.

Non-Product Line (NPL) Development: The NPL cost is calculated by adding the cost of non-product line development to software quality cost as follows:

$$\begin{aligned} C_{NPL}(N) &= N * [\text{LaborRate} * \text{COCOMO baseline}(100\ \text{KSLOC}) \\ &\qquad + [CD * (1 - TE) * \text{COQUALMO}(100\ \text{KSLOC}, EM_{NPL})]] \\ &= N * [\$8{,}000 * (2.94 * (100\ \text{KSLOC})^{1.0997} * 1.0)] + [\$10{,}000 * (1 - 0.9) * 1443.240] \\ &= N * \$\,5{,}165{,}762 \end{aligned}$$

Product Line (PL) Development: For simplicity, the COCOMO cost drivers in PL are identical to those used in NPL, with the exceptions that RUSE is Very High, DOCU is Very High, and RELY is Very High. Based on Equation 1, the cost of the product line is calculated as follows:

$$\begin{aligned} C_{PL}(N) &= C_{RCWR} + (N - 1) * C_{RCR} \\ &= [\$8{,}000 * (2.94\ (100\ \text{KSLOC})^{1.0997} * 1.0) * 1.469362) + (\$10{,}000 * (1 - 0.9) * 863.898)] \\ &\quad + (N - 1) * [\$8{,}000 * (2.94\,(50.11\ \text{KSLOC})^{1.0997} * 1.0) + (\$10{,}000 * (1 - 0.9) * 432.899)] \\ &= \$6{,}333{,}631 + (N - 1) * \$2{,}174{,}081 \end{aligned}$$

The saving of PL over NPL development is presented in Figure 2. Anywhere from one to five products are developed using NPL and PL development, and all include software quality costs. The first product is developed using RCWR product line development, to invest for future reuse. The remaining *N-1* products are developed using the RCR model, and benefit greatly from product line reuse.

The product line saves more money than NPL. These savings come from two sources: product line reuse and savings in software quality cost. After an initial product is developed, the product line, which reuses a portion of the initially developed product, reduces costs below that of standalone products. In addition, software qual-

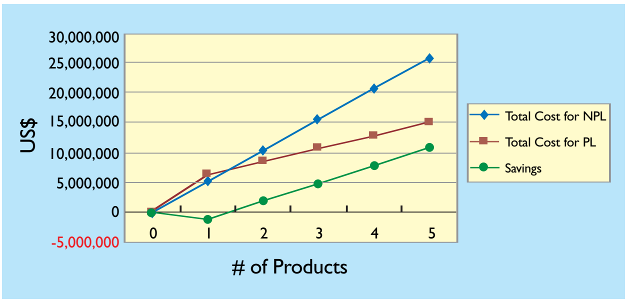

Figure 2. Saving of Non-Product Line (NPL) vs. Product Line (PL).

ity cost in product line development is much lower than that in NPL development of standalone products. The size of the product at each factory is reduced and the number of undetected defects is also reduced, due to reuse of some portion of the initially developed product.

To investigate the effect of software quality cost, the ROI figures calculated from qCOPLIMO (with software quality) are presented in Figure 3, and compared to COPLIMO (without considering software quality). This comparison reveals that the ROI based on COPLIMO or other related work is significantly underestimated if software quality cost is not considered.

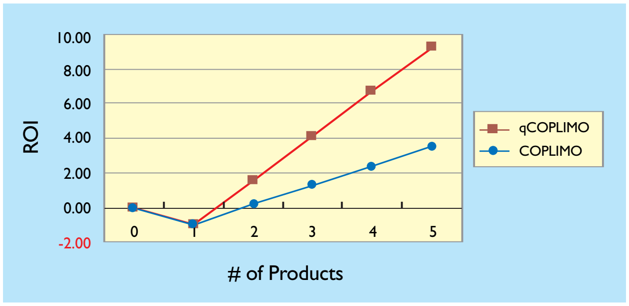

Figure 3. ROI Analysis of qCOPLIMO vs. COPLIMO.

Conclusion

The proposed qCOPLIMO provides a framework to estimate the effects of software quality cost for enabling cost-benefit analysis of software product lines. The majority of quantitative software product line models significantly underestimates the effect of software quality cost on potential savings and return on investment. These models only address development and life cycle costs. If these models consider the quality factor, however, they can have a considerably larger payoff by accumulating the potential savings of reusing components after removing product defects. **C**

References

1. Boehm, B., Abts, C., Brown, A.W., Chulani, S., Clark, B.K., Horowitz, E., Madachy, R., Reifer, D., and Steece, B. *Software Cost Estimation with COCOMO II*, Prentice Hall, 2000.
2. Boehm, B., Brown, A.W., and Yang, Y. A software product line life cycle cost estimation model. In *IEEE Proceedings of ISESE'04*, 2004.
3. Chulani, S., Boehm, B., and Steece, B. Bayesian analysis of empirical software engineering cost models. *IEEE Transactions on Software Engineering 25*, 4 (1999), 573–583.
4. Clements, P. and Northrop, L. *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2002.
5. CMU-SEI Product Line Practices Web site: www.sei.cmu.edu/activities/plp/.
6. Jacobson, I., Griss, M.L., and Jonsson, P. *Software Reuse*. Addison Wesley, 1997.
7. Lim, W. *Managing Software Reuse*. Prentice-Hall, 1998.
8. Poulin, J. *Measuring Software Reuse: Principles, Practices, and Economics Models*. Addison-Wesley, 1997.
9. Reifer, D.J. *Practical Software Reuse*. Wiley, New York, 1997.

Hoh Peter In (hoh_in@korea.ac.kr) is an associate professor in the College of Information and Communications at Korea University in Seoul, Korea.
Jongmoon Baik (jbaik@icu.ac.kr) is an assistant professor in the School of Engineering at Information and Communications University (ICU) in Daejeon, Korea.
Sangsoo Kim (sookim@korea.ac.kr) is a Ph.D. candidate in the College of Information and Communications at Korea University in Seoul, Korea.
Ye Yang (yangy@sunset.usc.edu) is a Ph.D candidate in the Center for Systems and Software Engineering at the University of Southern California in Los Angeles.
Barry Boehm (boehm@sunset.usc.edu) is TRW Professor of Software Engineering and the director of the Center for Systems and Software Engineering at the University of Southern California in Los Angeles.

This research was supported by the Ministry of Information and Communication (MIC), Korea, under the Information Technology Research Center (ITRC) support program supervised by the Institute of Information Technology Advancement (IITA) (IITA-2006-C1090-0603-0032).