Return on Investment of Software Product Line Traceability in the Short, Mid and Long Term

Zineb Mcharfi, Bouchra El Asri, Ikram Dehmouch, Asmaa Baya and Abdelaziz Kriouile
ENSIAS, Mohammed V Rabat University, Mohammed Ben Abdallah Regragui Avenue, Rabat, Morocco

Keywords: Traceability, Software Product Lines, Cost-benefits, Estimation Models, COPLIMO, METra-SPL.

Abstract: Several works discuss tracing in Software Product Lines from a technical and architectural points of view, by proposing methods to implement traceability in the system. However, before discussing this field of traceability, we first need to prove the profitability of integrating such approach in the Product Line. Therefore, we bring in this paper a quantitative analysis on how traceability can impact the Return on Investment of a Software Product Line, and in which conditions, in terms of number of products and SPL phase, can tracing be profitable. We compare the results of a generic Software Product Line estimation model, COPLIMO, and our model METra-SPL. Our analysis shows that introducing traceability costs when constructing the Product Line, but can be profit making in the long term, especially in maintenance phase, starting from 2 products to generate.

1 INTRODUCTION

Several studies highlight the importance of traceability in software engineering (Gotel et al., 2012). Even popular standards like CMMI (CMMI Product Team, 2006) incorporate this concept into their models and define requirements to effective traceability.

For large scale systems like Software Product Lines (SPL), tracing helps better know the system and facilitates its maintenance and evolution (Cavalcanti et al., 2011). Therefore, studies are conducted to best integrate traceability in SPL (Mäder and Gotel, 2012): implementation strategy, relations between artifacts, automation, maintenance, etc.

Despite this growing importance accorded to traceability in software engineering in general and SPL in particular, this concept is still rarely adopted in practice, as it is laborious, usually manual, time and resource consuming, and error prone (Ramesh and Jarke, 2001).

To better clarify the dilemma of cost and benefits of traceability in SPL, we decided to study the additional costs that can be generated when introducing a traceability approach in a SPL. Our analysis, as detailed thereafter, is based on a comparison between results obtained from on COPLIMO effort estimation model (Boehm et al., 2004), and our model METra-SPL (Metrics for Estimating Traceability in SPL), that takes into consideration additional elements.

The remainder of this paper is structured as follow: In Section 2 we introduce SPL and traceability in those large scale systems. In Section 3 we present traceability cost estimation related works, before detailing our proposed model METra-SPL and our comparative study in Section 4. We conclude and present further lines of research in section 5.

2 TRACEABILITY IN SOFTWARE PRODUCT LINES: A GROWING CHALLENGE IN A COMPLEX ENVIRONMENT

In this section we introduce traceability in SPL to present the motivations behind our present work.

2.1 Software Product Lines

As defined by Northrop (Northrop, 2002), a SPL is *"a set of software-intensive systems that share a common, managed feature set satisfying a particular market segment's specific needs or mission and that are developed from a common set of core assets in a*

Mcharfi Z., El Asri B., Dehmouch I., Baya A. and Kriouile A..
Return on Investment of Software Product Line Traceability in the Short, Mid and Long Term.
DOI: 10.5220/0005465304630468
In *Proceedings of the 17th International Conference on Enterprise Information Systems* (ICEIS-2015), pages 463-468
ISBN: 978-989-758-097-0
Copyright © 2015 SCITEPRESS (Science and Technology Publications, Lda.)

prescribed way". This approach is used in the organizations with massive production of products sharing the same components but answering specific needs. The common components (e.g., architecture, requirements, test plans, schedules, budgets and processes description) are called "core assets". Adopting a SPL approach allows to produce new systems by reusing the existing ones, in an organized manner.

SPL is a combination of three major interacting elements, called the SPL essential activities (Northrop, 2002; Northrop and Clements, 2005): (1) core asset development or Domain Engineering (DE), (2) product development or Activities Engineering (AE) and (3) technical and organizational management that orchestrates those two activities.

In such large scale system with many components to manage, stakeholders look for a guaranty that the products will meet the required quality and conformity to the initial requirements. Therefore, more proofs are needed, which can be achieved by implementing a traceability approach in the SPL.

2.2 Traceability in Software Product Lines

Traceability is an important element in software quality assurance: it allows producing and maintaining clear and consistent documentation, verifying that all requirements have been implemented (Cleland-Huang et al., 2014), and helps being independent from individual knowledge (Lindvall and Sandahl, 1996).

It is also an important element to decide on the architectural choices and facilitate communication between stakeholders (Anquetil et al., 2010). Traceability is very helpful when it comes to maintenance and evolution as it allows analyzing and controlling the impact of changes (Cavalcanti et al. 2011). This characteristic is very useful in SPL context where produced elements share a wild number of common components. Thus, once a product changes, traces help detecting other impacted products.

According to (Cleland-huang et al., 2012; Spanoudakis and Zisman, 2005; Ramesh and Jarke, 2001), traceability in SPL can be used either while developing, for short term purposes (e.g., to verify and validate requirements implementation), or in maintenance phase, for long term use (e.g., artifacts understanding, change management and components reuse). However, despite the objective of its use, many difficulties can be faced when implementing traceability in SPL (Jirapanthong and Zisman, 2005): (i) larger documentation than in traditional software development; (ii) heterogeneity of the documents; (iii) the need to link between different products and between them and the Product Line (PL) architecture. Also, unlike software engineering approaches for single systems, SPL introduces a complex dimension: variability. Variability represents an added difficulty to traceability in SPL as one needs to understand its consequences during the development phases (Jirapanthong and Zisman, 2005). Some works handle traceability and variability issues in SPL while tracing relations between artifacts (Anquetil et al., 2010; Anquetil et al., 2008; Berg et al., 2005), others manage it throw a metamodel for SPL development (Cavalcanti et al., 2011). Ghanam and Mauer (2009; 2008) use Acceptance Tests (AT) to generate test artifacts in an eXtreme Programing (XP) Agile SPL (ASPL) environment.

However, tracing in practice is laborious and its benefits can only be perceived in the mid to long term.

3 TRACEABILITY COST ESTIMATION RELATED WORKS

As discussed earlier, works that treat traceability issues are mostly interested in traceability strategy (relations between trace links, manual vs automated traceability, architecture, etc.). However, as traceability is rarely implemented in practice (Cleland-huang et al., 2012), there is need to prove its profitability compared to its complexity, in order to convince the project manager of the benefits of traceability implementation.

Therefore, some works on traceability cost estimation have been conducted, not only in the specific SPL traceability domain, but also at a larger scale, for software engineering in general, according to traceability strategy adopted. Egyed et al. (Egyed et al., 2005; Egyed, 2006; Egyed et al., 2005; Egyed et al., 2009) deal with the cost of trace links generation in an automated traceability approach. They demonstrate, through empirical studies, that a compromise can be made between acceptable trace quality and granularity, and low trace links costs. Still dealing with traces value, (Heindl and Biffl, 2005) present a study that takes into consideration many parameters to calculate the effort of tracing.

Those parameters are (i) number of artifacts to be traced; (ii) number of project requirements, as requirements traceability is an n² complexity problem; (iii) importance of each requirement from a stakeholder point of view; and (iv) requirements volatility, as frequently changing requirements are the ones that need most to be traced.

Few works deal with traceability implementation cost-benefits (Cleland-huang et al., 2012), and they generally rely on data post-analysis from already conducted case studies.

Next section describes our model METra-SPL and analysis the impact of tracing on SPL's implementation and maintenance ROI.

4 HOW TRACING CAN IMPACT SOFTWARE PRODUCT LINES RETURN ON INVESTMENT

In this section we present our contribution, the METra-SPL cost estimation model for SPL, which takes into consideration traceability costs and SPL phases.

We also compare COPLIMO and METra-SPL estimations for the same SPL case study to discuss tracing additional costs.

4.1 Our Cost Estimation Model: METra-SPL

COCOMO II is by far one of the most widely used cost estimation models. It is an adaptation of COCOMO 81 and has been adapted and declined in many specific models, like COPLIMO (Boehm et al., 2004), the COCOMO II derivation dedicated to SPL.

Our METra-SPL model is based on COPLIMO. It assumes the same bases but differ as it takes into consideration two important element that impacts SPL costs: tracing and SPL phases. In fact, tracing can generate additional costs as it requires links creation, adaptation and maintenance.

Also, SPL approach is based on two major processes: Domain Engineering (DE) and Application Engineering (AE). Consequently, trace links are generated while developing the PL, under DE process, and then used in AE process to identify artifacts relations and make the right decisions when choosing components to use while generating a product. In addition, trace links have an impact on SPL maintenance and change management: they help identify artifacts affected by changes and links have to be modified or created when updating the PL.

Taking into account those fundamentals, and regarding the different objectives between DE, AE and maintenance, we propose the METra-SPL model.

In this model, we decline the effort estimation equation into 3 expressions: the first one for DE, the second one for AE and the last one for maintenance effort estimation.

In fact, in DE, we implement specially developed products (PFRAC). They might be reused in AE (RFRAC), or adapted (AFRAC) and new ones integrated in the maintenance phase.

PMNR(N) represents the effort of developing N products under the SPL, which we adapt as follow:

- For DE effort estimation:

$$PMNR_{DE}(N) = N*A*(SIZE_{P})^{B}*DOCU*\prod(EM) \quad (1)$$

- For AE effort estimation:

$$PMNR_{AE}(N) = N*A*(SIZE_{R})^{B}*DOCU*\prod(EM) \quad (2)$$

- For maintenance estimation:

$$PMNR_{M}(N) = N*A*(SIZE_{P} + SIZE_{A})^{B} * \prod(EM) \quad (3)$$

$SIZE_{P}$, $SIZE_{R}$ and $SIZE_{A}$ are size of the specially developed product, reused products and adapted ones, respectively.

DE is based on products development, AE on products reuse, and maintenance on their adaptation and the development of new ones.

Also, as documentation is a principal element of traceability and environment implementation (either DE or AE), its impact has to be considered in measuring the SPL effort. Therefore, the impact of DOCU (Degree of Documentation) multiplier is made in evidence in (1) and (2). In (Boehm et al., 2004), the calibration multipliers ∏ (EM) takes the value 1, which means that all the multipliers are considered in their nominal value = 1. But in our case, the DOCU multiplier varies depending on traceability strategy instead of taking the same nominal value.

COPLIMO equations for calculating adapted SIZE and the Adaptation Adjustment Modifier (AAM) remain unchanged:

$$AAM_{A}=[AA+AAF*(1+(0,02*SU*UNMF))]/100 \quad (4)$$

AA, AAF, SU and UNMF are Assessment and Assimilation factor, Adaptation and Adjustment Factor, Software Understanding increment and the scale of Programmer's Unfamiliarity with the software, respectively.

- The SIZE of specific product development:

$$SIZE_P = PFRAC * SIZE \quad (5)$$

- For software portion to be reused:

$$SIZE_R = RFRAC * SIZE * AA/100 \quad (6)$$

- For software portion to be adapted:

$$SIZE_A = AFRAC * SIZE * AAM_A \quad (7)$$

- The resulting size (PSIZE) of each product is:

$$PSIZE = SIZE_P + SIZE_R + SIZE_A \quad (8)$$

4.2 SPL ROI: Tracing Impact in Short, Mid and Long Terms

In his estimation model COPLIMO (Boehm et al., 2004), Boehm presents a "macro" cost estimation of SPL development cycle, without taking into consideration tracing politic. Also, as SPL are reuse systems, the effort of producing an element can differ whether it is based on existing elements, adapted ones, or newly created artifacts. Therefore, COPLIMO as a generic model that gives a global SPL cost estimation.

In our METra-SPL model, we consider the impact of tracing on SPL development. We also distinguish, as described earlier, between different SPL processes: DE, AE and maintenance.

Let us consider a case study in an environment of aircraft-spacecraft production (Boehm et al., 2004). COPLIMO shows a positive SPL ROI starting from 3 products to develop. The same case study, but using METra-SPL estimation model, and taking into consideration SPL phases and tracing strategy (a targeted traceability in that case, where just efficient traces are created), shows some different results: As shown to Figure 1, tracing in DE costs. In fact, for an efficient tracing, one needs to adopt a traceability approach to identify efficient links, and use a support to store created links (e.g., tracing matrix). This can be time and effort consuming. However, we are still at the beginning of SPL implementation process.

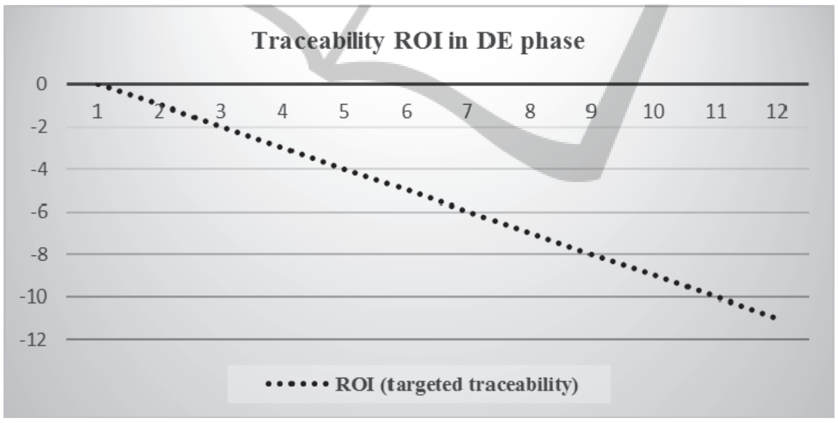

Figure 1: ROI in SPL DE based on METra-SPL.

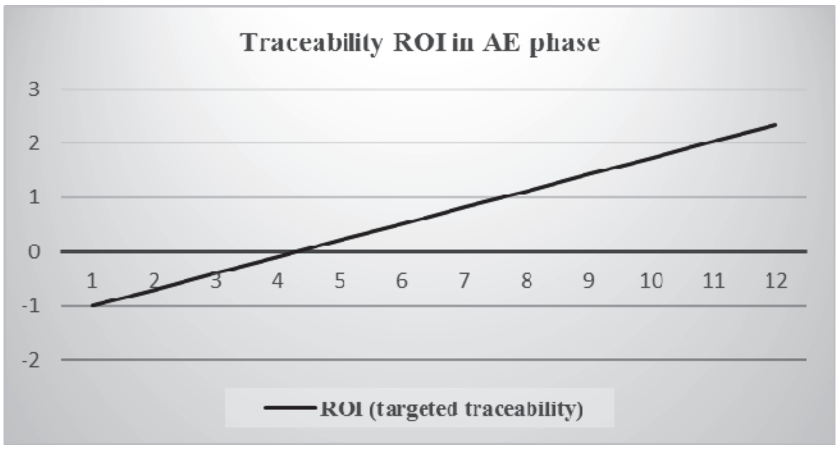

Figure 2: ROI in SPL AE based on METra-SPL.

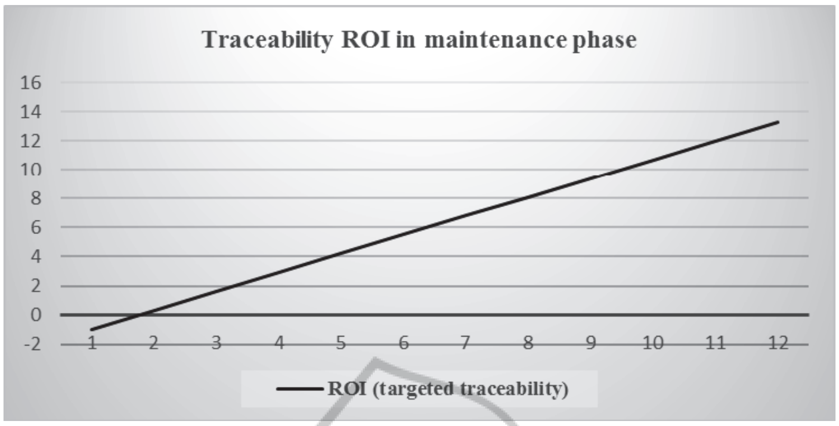

Figure 3: ROI in SPL maintenance phase based on METra-SPL.

In fact, in AE phase (Figure 2), with tracing, SPL ROI become positive starting from 5 products to generate. This is due to some new links to create, and the existing ones to adapt. Profitability of tracing is more visible in maintenance phase (Figure 3), where ROI already shows positive values for 2 products in the SPL: Maintenance is even easier when traceability is applied as the latter helps defining relations between different artifacts, and, consequently, change impacts are quickly and easily identified.

5 CONCLUSION

Despite the importance that studies accord to traceability in software engineering in general, implementing trace links in SPL is globally avoided as project managers are aware of initial tracing costs, but cannot quantitatively measure its benefits in the mid to long term (Cleland-huang et al., 2012).

Therefore, we conducted a study to quantify SPL ROI when introducing traceability in the short (DE), mid (AE) and long term (maintenance). This study was established based on an analysis of the METra-SPL model estimations, and comparison with COPLIMO results.

In general, METra-SPL allows a more detailed analysis of SPL ROI compared to COPLIMO. METra-SPL estimated values, however, are close to the generic one provided by COPLIMO.

Further studies can be conducted to study the impact of adopting a specific Traceability Information Model (TIM) (Mäder and Gotel, 2012) with automated link generation process, and possible new factors that may impact SPL ROI to be consequently introduced in our METra-SPL model.

REFERENCES

Anquetil, N. et al., 2010. A model-driven traceability framework for software product lines. *Software & Systems Modeling*, 9, pp.427–451.

Anquetil, N. et al., 2008. Traceability for Model Driven, Software Product Line Engineering. *ECMDA Traceability Workshop Proceedings*.

Berg, K., Bishop, J. & Muthig, D., 2005. Tracing Software Product Line Variability - From Problem to Solution Space. *Proceedings of the 2005 annual research conference of the South Africain institute of computer scientists and information technologists on IT research in developing countries*, pp.182–191.

Boehm, B. et al., 2004. A Software Product Line Life Cycle Cost Estimation Model.

Cavalcanti, Y.C. et al., 2011. Towards metamodel support for variability and traceability in software product lines. In *Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems - VaMoS '11*. New York, New York, USA: ACM Press, pp. 49–57.

Cleland-Huang, J. et al., 2014. Software Traceability: Trends and Future Directions. In *36th International Conference on Software Engineering (ICSE),*. Hyderabad, India.

Cleland-huang, J., Gotel, O. & Zisman, A., 2012. *Software and Systems Traceability*,

Cmmi Product Team, 2006. CMMI ® for Development , Version 1 . 2. , (August).

Egyed, A., Biffl, S., et al., 2005. Determining the Cost-Quality Trade-Off for Automated Software Traceability. In *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*. pp. 360–363.

Egyed, A., 2006. Tailoring software traceability to value-based needs. In *Value-Based Software Engineering*. Springer Berlin Heidelberg, pp. 287–308.

Egyed, A. et al., 2009. Value-Based Requirements Traceability : Lessons Learned. In *Design*

Requirements Engineering: A Ten-Year Perspective. pp. 240–257.

Egyed, A., Rey, M. Del & Grünbacher, P., 2005. A Value-Based Approach for Understanding Cost-Benefit Trade-Offs During Automated Software Traceability. In *The 3rd ACM Int. Workshop on Traceability in Emerging Forms of Software Engineering, TEFSE 2005, Held with the 20th IEEE/ACM Int. Conf. Automated Software Engineering, ASE2005*. ACM, pp. 2–7.

Ghanam, Y. & Maurer, F., 2008. An Iterative Model for Agile Product Line Engineering. *SPLC (2)*, pp.377–384.

Ghanam, Y. & Maurer, F., 2009. Extreme Product Line Engineering : Managing Variability and Traceability via Executable Specifications. In *2009 Agile Conference*. pp. 41–48.

Gotel, O., Cleland-Huang, J., Hayes, J. H., Zisman, A., Egyed, A., Grünbacher, P., Dekhtyar, A., Antoniol, G., Maletic, J. & Mäder, P., 2012. Traceability fundamentals. In *Software and Systems Traceability*. Springer London, pp. 3-22.

Heindl, M. & Biffl, S., 2005. A case study on value-based requirements tracing. *Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering - ESEC/FSE-13*, p.60.

Jirapanthong, W. & Zisman, A., 2005. Supporting product line development through traceability. In *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*. pp. 506–514.

Lindvall, M. & Sandahl, K., 1996. Practical Implications of Traceability. *Software: Practice and Experience*, 26(10), pp.1161–1180.

Mäder, P. & Gotel, O., 2012. Ready-to-Use Traceability on Evolving Projects. In Software and Systems Traceability. pp. 173-194

Northrop, L. & Clements, P., 2005. Software Product Lines. *Carnegie Engineering Institute*, pp.1–105.

Northrop, L. M., 2002. SEI's software product line tenets. *IEEE Software*, 19(4), pp.32–40.

Ramesh, B. & Jarke, M., 2001. Towards Reference Models for Requirements Traceability. *Software Engineering, IEEE Transactions on*, 27(1), pp.58–93.