De l'ombre à la lumière : plus de visibilité sur l'Eclipse

Boris Baldassari*, Flavien Huynh*, Philippe Preux**

* 76 Allées Jean Jaurès, Toulouse, France.
boris.baldassari@squoring.com – http://www.squoring.com
** Université de Lille 3, LIFL (UMR CNRS) & INRIA, Villeneuve d'Ascq, France
philippe.preux@inria.fr – https://sequel.inria.fr

Résumé. L'extraction de connaissances à partir de données issues du génie logiciel est un domaine qui s'est beaucoup développé ces dix dernières années, avec notamment la fouille de référentiels logiciels (Mining Software Repositories) et l'application de méthodes statistiques (partitionnement, détection d'outliers) à des thématiques du processus de développement logiciel. Cet article présente la démarche de fouille de données mise en œuvre dans le cadre de Polarsys, un groupe de travail de la fondation Eclipse, de la définition des exigences à la proposition d'un modèle de qualité dédié et à son implémentation sur un prototype. Les principaux concepts adoptés et les leçons tirées sont également passés en revue.

1 Introduction

L'évolution et le croisement des disciplines de la fouille de données et du génie logiciel ouvrent de nouveaux horizons pour la compréhension et l'amélioration du logiciel et des pratiques de développement. A l'aube de ce domaine émergent de nombreuses questions restent cependant d'actualité du point de vue de la conduite de programmes de mesure logicielle, notamment relevées par Fenton (1994) et Kaner et Bond (2004). Nous proposons dans cet article quelques pistes pour répondre à ces problématiques et mettre en place un processus de mesure fiable et efficace.

Il est utile pour la définition de la notion de qualité de s'appuyer sur des modèles ou standards reconnus : du point de vue de *la qualité produit*, la référence *de facto* semble être l'ISO 9126 et son futur successeur, la série 250xx SQuaRE. La *maturité du processus de développement* est adressée par des initiatives largement reconnues telle que le CMMi ou l'ISO 15504. Afin de clarifier la démarche de mesure, l'approche *Goal-Question-Metric* proposée par Basili et al. et reprise par Westfal Westfall et Road (2005) permet une approche plus rigoureuse, qui préserve l'efficacité de l'analyse et le sens de ses résultats.

2 Topologie d'un projet de développement logiciel

Chaque référentiel d'information utilisé par le projet possède des informations exploitables. Il importe de lister les référentiels disponibles, puis d'identifier pour chacun d'eux les artéfacts et mesures disponibles, et de leur donner un contexte sémantique.

Le code source est le type d'artéfact le plus utilisé pour l'analyse de projets logiciels. Du point de vue de l'analyse statique (Louridas, 2006), les informations que l'on peut récupérer d'un code source sont les *métriques*, correspondant à la mesure de caractéristiques définies du logiciel (e.g. sa taille ou la complexité de son flot de contrôle), et les *violations*, correspondant au non-respect de bonnes pratiques ou de conventions de codage ou de nommage (e.g. l'obligation de clause default dans un switch). Ces informations sont fournies par des analyseurs tels que Checkstyle, PMD ou SQuORE (Baldassari, 2012).

La gestion de configuration contient l'ensemble des modifications faites sur l'arborescence du projet, avec des méta-informations sur *l'auteur*, la *date* ou *l'intention* des changements. Le positionnement dans l'arbre des versions est important, car les résultats ne seront pas les mêmes pour une version en cours de développement (développée sur le tronc) et pour une version en maintenance (développée sur une branche).

La gestion des tickets recense l'ensemble des demandes de changement faites sur le projet. Ce peuvent être des problèmes (bugs), de nouvelles fonctionnalités, ou de simples questions.

Les listes de diffusion sont les principaux moyens de communication utilisés au sein de projets logiciels. Il existe en général au moins deux listes, une dédiée au développement et l'autre aux questions utilisateur.

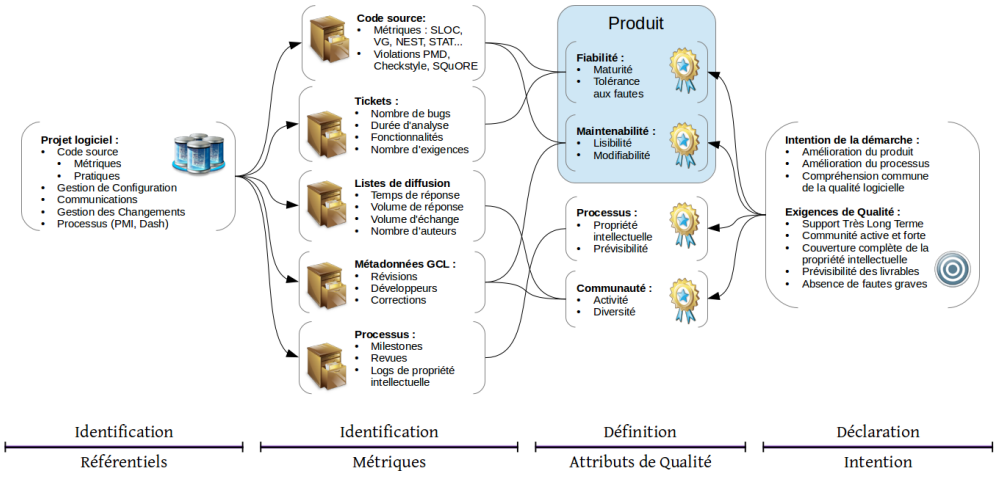

FIG. 1 – *Processus de définition du modèle de qualité et du choix des métriques.*

3 Présentation de la méthode d'extraction

La fouille de données logicielles est similaire en plusieurs points à la conduite de programmes de mesures. L'expérience glanée au fil des années sur ce dernier domaine (voir notamment Gopal et al. (2002) et Menzies et al. (2011)) a permis d'établir de bonnes pratiques qui aident à la définition, à l'implémentation et à l'utilisation des résultats d'analyse.

La déclaration d'intention donne un cadre méthodologique et sémantique pour la compréhension et l'interprétation des résultats. Par exemple, les mesures accessibles et les résultats attendus ne seront pas les mêmes pour un audit et pour un programme d'amélioration continue de la qualité. L'intention doit être *simple* et *tenir en quelques phrases*.

La décomposition des attributs de qualité décompose les objectifs de la fouille et forme le lien entre une déclaration informelle d'intention et les mesures concrètes qui vont permettre de mesurer et d'améliorer la qualité ainsi définie ; elle doit faire l'objet d'un consensus général et être validée par les acteurs du programme.

La définition des métriques accessibles à partir des référentiels identifiés sur le projet doit être *fiable* – i.e. l'information recherchée est systématiquement présente et valide – et *compréhensible* pour garder la confiance des acteurs dans le processus.

L'implémentation du processus de collecte et d'analyse doit être *transparente* pour que les acteurs puissent se l'approprier, intégralement *automatisée*, et *exécutée de manière régulière*.

La présentation des informations est capitale. Dans certains cas une liste concise d'artéfacts est suffisante, alors que dans d'autres cas un graphique bien choisi sera plus adapté et délivrera en quelques secondes l'essentiel du message.

4 Mise en pratique avec Eclipse

Cette approche a été mise en œuvre dans le cadre de Polarsys, un groupe de travail de la fondation Eclipse qui a pour but, entre autres, de proposer un cadre d'évaluation de la qualité des projets de la fondation. L'arbre de qualité montré en figure 2 montre les exigences identifiées pour Eclipse et Polarsys, et leur organisation en attributs de qualité.

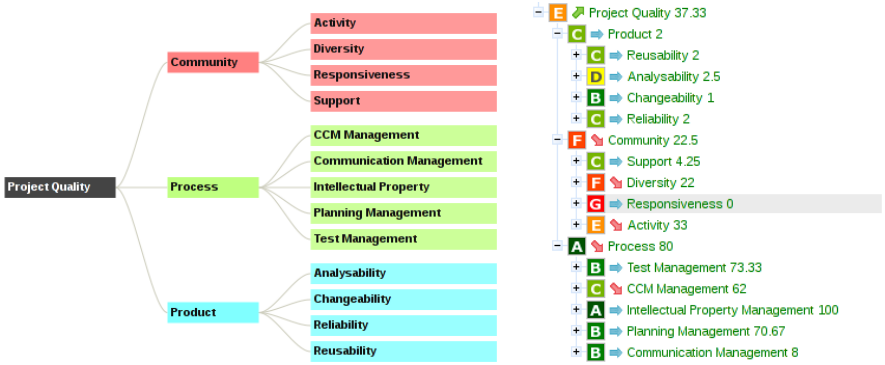

FIG. 2 – *Modèle de qualité proposé pour Polarsys.*

Du point de vue de **l'aide à la prise de décision**, les axes de qualité choisis permettent d'évaluer la maturité du projet immédiatement, selon les critères propres à l'organisation. Par exemple le projet analysé sur la droite de la figure 2 montre une bonne qualité produit, mais une faible activité de communication et un petit nombre de participants, car l'essentiel des contributions est fait par une unique société. Cette caractéristique peu visible *a priori* en fait un risque pour la pérennité du projet, rapidement identifiable sur l'arbre de qualité. Du côté

des équipes de développement, **l'aide à l'amélioration de la qualité** se fait au moyen de listes de violations et de points d'action, qui permettent d'identifier et d'améliorer les pratiques non acquises et fournissent des recommandations pragmatiques pour l'amélioration du code et du processus.

5 Conclusion

Le prototype développé dans le cadre du groupe de travail Polarsys a permis de **définir un socle méthodologique commun** et **un ensemble de métriques** issues de sources nouvelles pour travailler ensemble sur la notion de qualité, et a plus généralement permis de **démontrer la faisabilité d'une telle solution**. Le prototype a apporté la preuve que des connaissances pratiques pouvaient être extraites du projet pour l'évaluation et l'amélioration de la maturité. Ce travail a été présenté à la communauté Eclipse lors de la conférence EclipseCon France 2013 sise à Toulouse en Juin 2013, et son industrialisation est en cours pour une publication prévue en 2014. Il est également envisagé d'étendre la démarche de qualité initiée pour Polarsys aux projets Eclipse pour compléter les initiatives actuelles PMI et Dash.

Références

Baldassari, B. (2012). SQuORE : A new approach to software project quality measurement. In *International Conference on Software & Systems Engineering and their Applications*.

Fenton, N. (1994). Software Measurement : a Necessary Scientific Basis. *IEEE Transactions on Software Engineering 20*(3), 199–206.

Gopal, A., M. Krishnan, T. Mukhopadhyay, et D. Goldenson (2002). Measurement programs in software development : determinants of success. *IEEE Transactions on Software Engineering 28*(9), 863–875.

Kaner, C. et W. P. Bond (2004). Software engineering metrics : What do they measure and how do we know ? In *10th Internationalwestfal Software Metrics Symposium*, pp. 1–12.

Louridas, P. (2006). Static Code Analysis. *IEEE Software 23*(4), 58–61.

Menzies, T., C. Bird, et T. Zimmermann (2011). The inductive software engineering manifesto : Principles for industrial data mining. In *Proceedings of the International Workshop on Machine Learning Technologies in Software Engineering*, Lawrence, Kansas, USA.

Westfall, L. et C. Road (2005). 12 Steps to Useful Software Metrics. *Proceedings of the 17th Annual Pacific Northwest Software Quality Conference 57 Suppl 1*(May 2006), S40–3.

Summary

Knowledge discovery and extraction from software-related repositories had great advances in recent years. In this article, we present a data mining program initiated within the Eclipse foundation to assess and improve software project maturity. We thoroughly describe the approach and the key concepts retained. Results are discussed and future directions proposed to foster further work in the domain.