SQuORE : une nouvelle approche de mesure de la qualité des projets logiciels

Boris Baldassari,

SQuORING Technologies, 76 Allées Jean Jaurès, 31000 Toulouse
Tel: +33 581 346 397, E-mail: boris.baldassari@squoring.com

Abstract : La qualité a un prix. Mais la non-qualité est encore plus chère. Connaître les coûts et la valeur du patrimoine logiciel, mieux comprendre les équipes de développement, ou évaluer rapidement la qualité d'un développement externalisé sont des activités centrales pour toute société dont le fonctionnement s'appuie sur l'informatique. Des standards et outils ont été mis au point pour adresser ces éléments, avec plus ou moins de succès, mais les difficultés sont nombreuses, de par la diversité des projets logiciels, la mise en place effective d'un processus de mesure – de la sélection des objectifs et métriques de mesure à l'affichage de l'information, ou la compréhension par les acteurs du processus qualité. C'est pour ces situations que l'outil SQuORE introduit une approche novatrice de la qualité logicielle, plus fiable, plus intuitive et plus contextuelle, appliquant les techniques de la Business Intelligence à l'analyse des projets logiciels. SQuORE offre ainsi une vision et une compréhension unifiées de l'état et des avancées de projets, permettant une amélioration pragmatique du processus et des développements. Cet article présente comment la solution SQuORE résout le dilemme de la qualité, et décrit deux cas issus de projets industriels : un programme de priorisation des tests unitaires, et un modèle de processus plus complet, s'appuyant sur de nombreux éléments hors code source.

1 Introduction

Malgré un intérêt croissant pour les notions de qualité logicielle, beaucoup la considèrent encore comme une activité coûteuse et non productive. Mary Shaw[23] a étudié en 1990 l'évolution de la discipline du génie logiciel, en la comparant avec d'autres sciences plus anciennes et plus matures (telles que l'architecture) ; le génie logiciel n'en est qu'à ses premiers pas en tant que science, et la discipline doit encore évoluer et gagner en maturité pour parfaire son développement. Mais de

nombreux progrès ont été faits ces dernières années, l'expérience s'accumule et l'heure est venue d'une nouvelle ère pour le développement logiciel, avec l'analyse multidimensionnelle [12] et la prise de décision par fouille de données [5].

Dans la seconde section de cet article, nous rappelons quelques termes et définitions, et récapitulons l'état de l'art et des pratiques actuelles. La section 3 décrit l'approche choisie pour SQuORE, ses fonctionnalités et avantages. La section 4 étend davantage le périmètre de l'analyse logicielle en montrant deux exemples industriels d'implémentation de modèles de qualité, pour les tests et pour la gestion de projet.

2 Etat de l'art

1 Coût de la non-qualité

Les exemples d'erreurs logicielles et leurs coûts associés ne manquent pas, que les pertes soient humaines ou financières. Toutes trouvent leur origine dans un manque de qualité et de contrôle sur le processus de développement. Quelques exemples connus sont reproduits ci-après :

- Therac-25 : surdosage de radiations sur un appareil médical : 6 morts et deux ans avant correction [15].
- Marc Climate Observer : durée des tests insuffisante, le robot s'est écrasé suite au remplissage de son espace mémoire [19].
- Missile Patriot : dérive de 6 mètres par heure due à un bug de conversion. 28 soldats tués à Dhahran [25].
- Ariane 5 : réutilisation abusive de routines écrites pour Ariane 4, 500 M$ de pure perte [16].
- Crash du réseau AT&T (90's) : une mise à jour des routeurs rend le réseau inutilisable pendant 9 heures, 60M$ de perte de revenus, cause du bug : un break manquant [20].

Le National Institute for Standards and Technology (NIST) a commandé en 2002 un rapport sur le coût estimé des problèmes logiciels pour l'industrie américaine : 60 Milliard de dollars par an. Le rapport technique CHAOS, édité par le Standish Group, faisait état en 2004 de taux d'échec des projets logiciels de 70%. Les chiffres sont un peu meilleurs depuis, mais le constat reste accablant.

2 Standards pour produits et processus logiciels

De nombreux standards et normes dédiés à la qualité ont été produits par les organismes de standardisation ces dernières décennies, certains adressant plutôt la qualité des produits (de Boehm [1] à McCall [18], simplifié et amélioré par l'ISO 9126 [9]), tandis que d'autres adressent plutôt la qualité du processus (ainsi ISO 15504 [6] et le CMMi [2]). Plus récemment, deux nouveaux modèles ont vu le jour : SQALE [13, 14], une méthode indépendante du langage et des outils d'analyse utilisés, qui s'appuie principalement sur les notions de dette technique et de design, et l'ISO SQuaRE [7], successeur de l'ISO 9126 toujours en cours de développement. De plus, certains domaines ont leur propre standards de-facto : HIS et ISO 26268 [8] pour l'industrie automobile ou

la DO-178 [22] pour l'aéronautique et les systèmes embarqués critiques.

Mais certaines objections à ces modèles ont vu le jour dans les dernières années :

- Les standards, et certains modèles en particulier, sont parfois considérés comme le jugement subjectif et l'opinion d'experts, ainsi que relevé par Jung et al. [10] pour l'ISO 9126.
- Ils ne sont pas adaptés à tous les contextes et tous les besoins de qualité [11].

Un autre argument employé, toujours dans le cadre de la complexité et de la diversité des projets logiciels, est qu'ils ne proposent aucune mesure, ni aucun outil pratique pour mettre en œuvre l'analyse de qualité, ce qui provoque l'incompréhension de certaines mesures – à titre d'exemple, considérons la métrique MTBF, ou Mean Time Between Failure et la myriade de définitions qui lui sont associées [12].

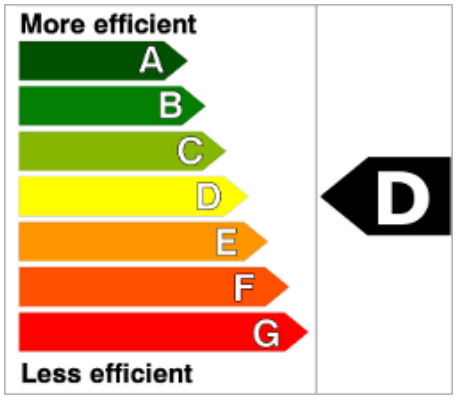
3 Métriques pour la mesure de la qualité logicielle

La littérature propose un grand nombre de métriques orientées logiciel, telles que par exemple la complexité cyclomatique, introduite par McCabe [17] pour mesurer la complexité du flot de contrôle, ou les mesures d'Halstead [4], qui permettent de qualifier le flot de données.

Un autre moyen d'évaluer la qualité d'un code est le nombre de non-conformités à une certaine référence. Par exemple, si des conventions de nommage et de codage ont été décidées, toute violation à ces règles est supposée diminuer la qualité, ou certaines de ses sous-caractéristiques, telles que l'analysabilité (pour les conventions de nommage) ou la fiabilité (pour les conventions de codage).

La notion de dette technique, édictée la première fois par Ward Cunningham en 1992 [3], et suscitant de plus en plus d'intérêt de nos jours, peut être vu comme la distance entre l'état actuel du logiciel et l'état de qualité souhaité. En ce sens, la dette technique est largement liée au nombre de non-conformités présentes dans l'application.

La tendance actuelle de la qualimétrie va vers l'analyse multidimensionnelle [12] : la qualité d'un produit ou d'un processus logiciel est la composée d'un ensemble de caractéristiques ou attributs, et donc de leurs mesures associées. Reste à savoir comment agréger et consolider ces mesures composites.



3 Les principes de SQuORE

En quelques mots, SQuORE récupère des informations issues de sources variées, agrège *intelligemment* les données et présente un rapport optimisé de l'état du logiciel ou du projet. La note principale de l'application est affichée sur une échelle de 7 niveaux, reconnaissable par tous, et donnant une indication visuelle immédiate sur les résultats de l'analyse.

Figure 1 : La notation SQuORE

4 Architecture

Le processus d'analyse SQuORE se décompose en trois étapes distinctes : la récupération des données par les data providers, le calcul des données composées par le moteur (engine), et la présentation des résultats, courbes et tableaux : le dashboard.

5 Data Providers

Ainsi que dit précédemment, il existe aujourd'hui de nombreux outils d'analyse de la qualité, chacun avec son propre domaine d'expertise et apportant une vue intéressante, mais partielle, sur la qualité. SQuORE rassemble ces informations et apporte la cohérence nécessaire à une utilisation intelligente de ces données. Tous les types de données sont acceptés : XML, CSV, appels API, jusqu'aux formats binaires, tels les anciens documents Microsoft Word.

L'analyseur SQuORE est exécuté en premier ; il est rapide, ne demande aucune compilation ou dépendance, et construit l'arbre des artefacts (quel que soit leur type : code source, tests, documentation, composants hardware, etc.) sur lesquels viendront se greffer les mesures issues d'outils tiers.

6 Consolidation des données

Une fois les mesures de base collectées, SQuORE agrège l'ensemble et calcule les mesures dérivées définies dans le modèle qualité pour chaque artefact, ainsi que montré en Figure 2.

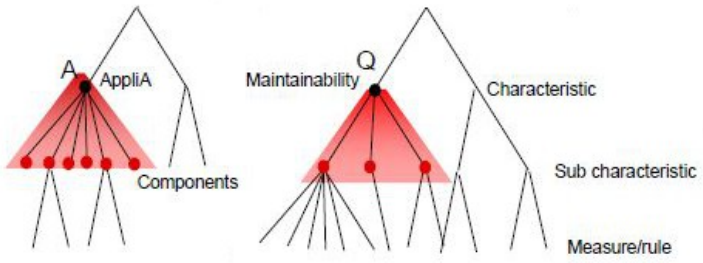


Figure 2 : Arbres des artefacts et de qualité

Modèles de qualité

Les modèles de qualité définissent la manière dont sont agrégées les métriques, des feuilles (fonctions, tests, etc.) jusqu'à la racine du projet : les mesures composites sont associées à un attribut de la qualité, lui-même affecté à chaque artefact de l'arbre.

Il n'existe pas de modèle qualité parfait et universel : chacun doit adapter la méthode d'analyse, en prenant en considération les besoins spécifiques et les objectifs de développement. Dans la plupart des cas l'élaboration du modèle de qualité se fait à partir de l'un des modèles existants, qui est adapté, pondéré différemment en fonction du contexte. Pour des situations plus complexes SQuORE propose tous les moyens nécessaires à la mise en œuvre d'un nouveau modèle, des calculs les plus simples aux requêtes les plus alambiquées.

Métriques, échelles, indicateurs

Les mesures en elles-mêmes donnent la valeur d'une caractéristique, sans jugement qualitatif. Les indicateurs fournissent cette information en comparant la valeur à une échelle de niveaux adaptée, ce qui permet de configurer les seuils et poids pour chaque niveau de jugement. Par exemple, le célèbre nombre cyclomatique [17] au niveau des fonctions pourrait être réparti sur une échelle de 4 niveaux :

- De 1 à 7, la fonction est notée A (très bon) et son poids de dette technique est de 0.
- De 8 à 15, la fonction est notée B (ok), et son poids est de 2.
- De 16 à 25, la fonction est notée C (mauvais) et son poids est de 4,
- Au-delà de 25, la fonction est notée D (très mauvais) et son poids est de 16 – car un refactoring est nécessaire.

Appliqué à cette échelle, l'indicateur du nombre cyclomatique donne immédiatement l'état de complexité de la fonction, quels que soient les triggers utilisés pour le contexte local.

Action Items

Il arrive souvent qu'une caractéristique que nous souhaitons évaluer dépende de plusieurs mesures de base. Par exemple, si une fonction est plutôt longue, a une complexité au-dessus de la moyenne, de nombreuses violations et un faible taux de commentaire, il est certainement intéressant de s'y pencher, alors même que chacun de ces éléments, pris individuellement, ne sont pas décisifs. Les action items permettent de mettre en place des triggers qui se déclenchent sur une variété de critères entièrement paramétrables. Une description du problème et des pistes pour sa résolution sont affichées, avec l'emplacement précis des différents critères qui ont déclenché le trigger. Les triggers peuvent être positionnés sur tout type d'artefact, ce qui fait d'eux un moteur puissant, par exemple pour formaliser et automatiser des heuristiques issues de l'expérience et du savoir-faire des équipes.

7 De la qualité produit à la gestion de projets

L'évaluation de la qualité d'un produit ou d'un processus est la première étape pour mieux contrôler ses développements (si l'on ne sait pas où l'on est, une carte est de peu d'utilité). L'étape suivante est de surveiller l'évolution du projet au fil du temps et des versions, et d'en extraire les éléments nécessaires à la prise de décision – processus décrit par Hassan et al. [5]. SQuORE propose de nombreux mécanismes d'exploration pour aider à cette démarche :

- Les graphiques d'évolution montrent l'historique des mesures, et attributs de qualité au fil des versions, et la tendance.
- Les nombreuses options de filtre et de tri permettent de localiser rapidement certains éléments (e.g. éléments dégradés, nouvelles fonctions, fichiers les plus mal notés).
- L'arbre de qualité montre quelles caractéristiques de la qualité se sont dégradées : maintenabilité du code, dérive du planning, tests échoués ?
- Les actions items permettent d'identifier des schémas d'évolution complexes, basés sur des caractéristiques actuelles ou sur certaines tendances (par exemple, dégradation de la testabilité, échecs nombreux sur les tests existants, diminution du taux de couverture de test).

4 Cas d'utilisation

8 Retours immédiats des équipes

Lors de nos présentations de l'outil SQuORE, certaines situations et réactions reviennent régulièrement:

- Les personnes sont capables d'identifier rapidement certains problèmes sérieux, sans être submergées par le flot d'informations. Souvent, ces problèmes avaient déjà été détectés par d'autres outils, mais l'information n'avait pas été identifiée ou suffisamment mise en valeur.
- Le sentiment général à propos de certaines applications connues des équipes est confirmé par des preuves factuelles et démontrables. Cela démontre la représentativité de la mesure et rend visibles et argumentables les pratiques de développement.
- Les développeurs sont concernés par la note de leur code et sa représentation simple et concise. L'utilisation de l'échelle d'énergie, reconnue de tous, renforce le caractère de standard (local) du modèle de qualité.

9 Modèle de test unitaire

Les délais de développement étant toujours trop courts pour le marché, il arrive parfois de devoir prioriser l'effort de test pour maximiser la rentabilité et le nombre de problèmes trouvés avant livraison, sachant que l'exécution complète des tests est de toute façon impossible dans les temps impartis.

L'un de nos clients avait seulement quelques jours de test pour qualifier une version logicielle. Jusque-là, le processus était essentiellement humain : les fichiers à tester étaient sélectionnés par rapport à leur historique de test, leurs récents changements, la complexité des fonctions et le nombre de violations détectées. A force d'expérience, certaines heuristiques avaient été développées pour aider à l'identification des fichiers prioritaires, avec attributs et seuils de tolérance.

Nous avons donc construit un modèle de qualité dédié, prenant en entrées les sorties de deux outils utilisés par l'équipe : QAC et Rational Test Real Time, plus l'analyseur SQuORE. Quatre mesures ont été définies : nombre de non-conformités QAC, RTRT, SQuORE, et le nombre cyclomatique de chaque fonction. La notation du modèle est construite sur une base exponentielle. Par exemple, si un fichier a une seule de ses mesures en défaut, sa note de dette technique sera de 2. Si deux, trois, ou quatre de ces mesures sont en défaut, le fichier recevra une note qui sera respectivement de 8, 16, 32. Cette structure permet d'identifier rapidement les pires fichiers.

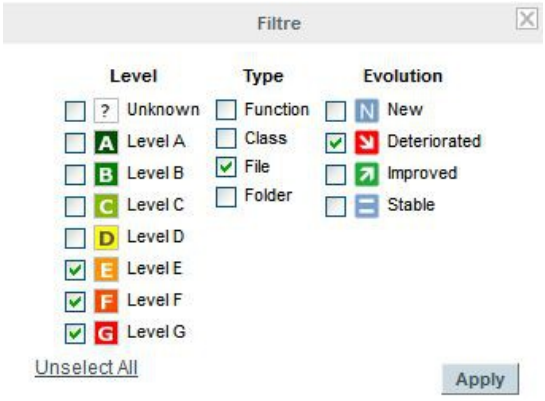


Figure 3 : Filtres d'artefacts

La notation des répertoires est basée sur le ratio de « mauvais » fichiers récursivement trouvés sous ledit répertoire, afin de mettre en relief les pires composants du logiciel simplement en triant l'arbre des artefacts par rapport à la notation de chaque répertoire.

Des action items spécifiques ont été mis en place pour les fichiers présentant un risque réel pour la validation, parce qu'ils ont une complexité cyclomatique réellement importante, ont un nombre de violations inquiétant ou une très mauvaise notation. Ces triggers permettent notamment de trouver de mauvais éléments cachés dans des hiérarchies de « bons » fichiers ou composants.

Nous avons ainsi pu :

- Mettre en place un modèle de prédiction d'efforts de test formalisé, et reconnu par les acteurs comme étant efficace. Le temps gagné à ce moment du processus est capital pour la mise sur le marché, et permet d'exécuter davantage de tests, puisque la détection est plus rapide.
- Reproduire et automatiser une heuristique « au ressenti » qui était auparavant affaire d'expérience et de savoir-faire. En le formalisant et en l'automatisant, les équipes ont pu davantage se concentrer sur l'effort de test.

10 Suivi de projet

Une autre expérience industrielle concernait la mise en place d'un suivi de projet plus complet, avec progression du backlog, avancées des tests et surveillance de la qualité logicielle.

L'analyseur SQuORE a été utilisé pour la qualité produit (code Java), et des data providers ont été mis en place pour récupérer les informations des outils de gestion des changements, test, et planning.

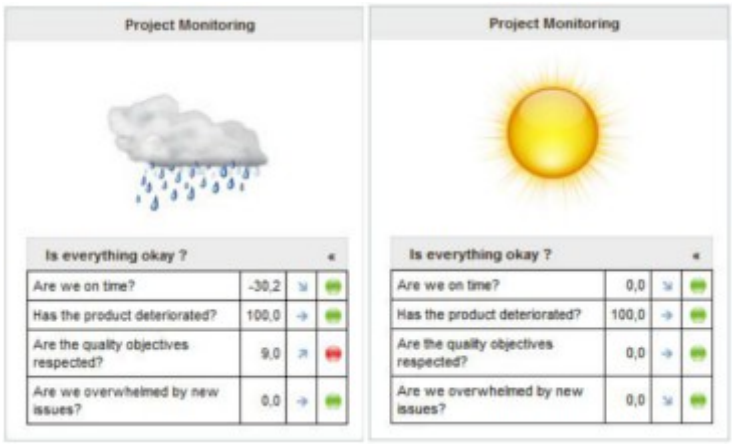


Figure 4: Scorecards de suivi de projet

4 axes de qualité ont été définis par rapport à ces entrées, résumés par les questions suivantes (cf. l'approche de V. Basili : Goal-Question-Metric [26]).

- « Est-ce que les objectifs de qualité sont respectés ? », basé sur la norme ISO9126.
- « Sommes-nous dans les temps? », utilisant les notions agiles de tâches.
- « Sommes-nous surchargés par les nouvelles requêtes ? », avec le nombre de bugs reportés en attente de traitement.
- « Le produit s'est-il détérioré ? », qui reprend les résultats de test (couverture et exécution).

Ces axes sont ramenés sur un unique indicateur, montrant l'avancement global du projet. Dans ce cas, la pire note trouvée parmi les sous-caractéristiques est utilisée en note globale, car il est considéré comme nominal que chacun de ces indicateurs soit positif.

Nous avons développé des action items afin d'identifier :

- Les parties du logiciel (composants ou fichiers) qui ont une mauvaise note de

maintenabilité, sont peu couverts par les tests, et sont concernées par beaucoup de reports de bugs.

- Les dérives de planning, lorsque le nombre de tâches ouvertes est trop important et de nombreux bugs sont ouverts.

L'automatisation des opérations de récupération des données, puis d'exécution de l'analyse et de publication des résultats est un élément clef de la réussite d'un projet de qualimétrie [27] ; Jenkins a donc été mis en place comme serveur d'intégration continue pour exécuter SQuORE régulièrement (exécutions quotidienne et hebdomadaire).

Des dashboards spécifiques ont été mis en place pour suivre aisément les critères importants de la gestion de projet : maintenabilité, requêtes ouvertes, tâches en attente, tests exécutés avec succès... Les informations de planning ont été représentées par les graphiques agiles de burn-up et burn-down.

Figure 5 : Arbre de qualité suivi de projet

Nous avons ainsi pu :

- Mettre en place un suivi quotidien et consolidé de l'état d'avancement des développements, avec surveillance des régressions et de la qualité.
- Reprendre le contrôle du projet et redonner confiance aux acteurs, managers et développeurs, en proposant une vision claire et concise, partagée et cohérente des développements.

5 References

[1] B. W. Boehm, J. R. Brown, and M. Lipow. Quantitative evaluation of software quality. In *Proceedings of the 2nd international conference on Software engineering* , pages 592-605, San Francisco, California, United States, 1976. IEEE Computer Society Press.

[2] CMMI Product Team. CMMI for Development, Version 1.3. Technical report, Carnegie Mellon University, 2010.

[3] Martin Fowler. Martin Fowler on Technical Debt, 2004.

[4] Maurice H. Halstead. *Elements of Software Science (Operating and programming systems series)*. Elsevier Science Inc., New York, NY, USA, 1977.

[5] Ahmed E Hassan and Tao Xie. Software Intelligence: The Future of Mining Software Engineering Data. In *Proc. FSE/SDP Workshop on the Future of Software Engineering Research (FoSER 2010)*, pages 161-166. ACM, 2010.

[6] ISO IEC. ISO/IEC 15504-1 – Information Technology – Process assessment. *Software Process: Improvement and Practice*, 2(1):35{50, 2004.

[7] ISO IEC. ISO/IEC 25000 – Software Engineering – software product quality requirements and evaluation (SQuaRE) – guide to SQuaRE. *Systems Engineering*, page 41, 2005.

[8] ISO. ISO/DIS 26262-1 - Road vehicles Functional safety Part 1 Glossary. Technical report, International Organization for Standardization / Technical Committee 22 (ISO/TC 22), 2009.

[9] ISO/IEC. ISO/IEC 9126 – *Software Engineering – Product quality*. 2001.

[10] Ho-Won Jung, Seung-Gweon Kim, and Chang-Shin Chung. Measuring software product quality: A survey of ISO/IEC 9126. *IEEE Software*, 21:88-92, 2004.

[11] Stephen H. Kan. *Metrics and Models in Software Quality Engineering*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2002.

[12] C Kaner andWalter P. Bond. Software engineering metrics: What do they measure and how do we know? *In 10Th International Software Metrics Symposium, METRICS 2004*, pages 1-12, 2004.

[13] Jean-louis Letouzey. The SQALE method for evaluating Technical Debt. In 2012 Third International Workshop on Managing Technical Debt, pages 31-36, 2012.

[14] Jean-louis Letouzey and Thierry Coq. The SQALE Analysis Model An analysis model compliant with the representation condition for assessing the Quality of Software Source Code. In *2010 Second International Conference on Advances in System Testing and Validation Lifecycle (VALID)*, pages 43-48, 2010.

[15] Nancy Leveson and Clark S. Turner. An Investigation of the Therac-25 Accidents. *IEEE Computer*, 26(7):18{41, 1993.

[16] J.L. Lions. ARIANE 5 Flight 501 failure. Technical report, 1996.

[17] TJ McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, (4):308-320, 1976.

[18] J.A. McCall. *Factors in Software Quality: Preliminary Handbook on Software Quality for an Acquisition Manager*. *Information Systems Programs, General Electric Company*, 1977.

[19] National Aeronautics and Space Administration. Mars Climate Orbiter Mishap Investigation Report. Technical report, National Aeronautics and Space Administration, Washington, DC, 2000.

[20] Peter G. Neumann. Cause of AT&T network failure. *The Risks Digest*, 9(62), 1990.

[21] National Institute of Standards and Technology (NIST), Department of Commerce. Software errors cost U.S. economy $59.5 billion annually, 2002.

[22] RTCA. DO-178B: Software Considerations in Airborne Systems and Equipment Certification. Technical report, Radio Technical Commission for Aeronautics (RTCA), 1982.

[23] Mary Shaw. Prospects for an engineering discipline of software. *IEEE Software*, (November):15-24, 1990.

[24] The Standish Group International Inc. The Standish Group International Inc. Chaos Technical report. Technical report, 2004.

[25] United States General Accounting Office. Patriot Missile Defense: Software Problem Led to System Failure at Dhahran, Saudi Arabia. Technical report, United States General Accounting Office, 1992.

[26] Basili, V. R., Caldiera, G., & Rombach, H. D. (1994). The goal question metric approach. Encyclopedia of Software Engineering. Wiley.

[27] Basili, V., & Weiss, D. (1984). A methodology for collecting valid software engineering data. IEEE Trans. Software Eng., 10(6), 728-738.