Squash, un modèle d'évaluation de la qualité des systèmes d'informations

Karine Mordal*

*Docteur en informatique à l'université Paris 8
2 rue de la liberté, 93526 Saint-Denis Cedex
kmordal@gmail.com,

Résumé. A partir de l'étude des normes ISO 9126 et SQuaRE notamment, nous avons formalisé un modèle de qualité nommé Squash, prenant en compte les particularités et les exigences du domaine industriel. Nous avons tout d'abord entrepris de poser une définition de la qualité et de définir les propriétés que nous voulions voir respecter par notre modèle en nous basant à la fois sur les modèles existants mais également sur les exigences des entreprises, partenaires de nos travaux de recherche. Puis nous avons déterminé la liste des propriétés indispensables à une bonne évaluation de la qualité que nous avons appliqués à nos différents modes de calcul : évaluation à partir de métriques ou de données non issues de métriques. A partir de ces propriétés nous avons construit un modèle de qualité appelé Squash, qui permet d'adapter l'évaluation de la qualité logicielle en fonction des systèmes et des exigences liées au système.

1 Introduction

Dans la pratique on constate qu'une démarche d'évaluation de la qualité est souvent initiée de deux manières différentes. Soit les différents acteurs de la conception logicielle se retrouvent confrontés à des problèmes qui leur sont propres et tentent de les résoudre de manière indépendante, soit la démarche est initiée au niveau hiérarchique supérieur. Dans le premier cas, les outils utilisés diffèrent souvent selon les métiers : un développeur optera le plus souvent pour des solutions lui donnant des mesures brutes sur son code source (par exemple l'outil Sonar) ; un testeur de son côté fera appel à des outils lui permettant de gérer les bases de tests et d'obtenir des métriques associées à ces tests (par exemple Salomé [1] ou TestLink [2]). Dans le second cas, lorsque la démarche qualitative est envisagée dans une démarche liée à l'entreprise elle-même et non plus seulement au cas particulier d'une application, les *managers* souhaitent avoir une vue globale de cette qualité et non pas une série de résultats de métriques. Il font habituellement appel à des modèles de qualité (tels que la norme ISO 9126) pour obtenir une évaluation d'ensemble de la qualité. Ces modèles reposent sur les mesures et les analyses des différents corps de métier pour leur donner sens à un niveau plus global : ils tentent d'évaluer la qualité de manière plus générale. Il s'agit alors d'appliquer une démarche homogène et

1. https ://wiki.ow2.org/salome-tmf/
2. http ://www.teamst.org

d'utiliser un référentiel commun pour évaluer non plus une application ou encore une partie de l'application, mais d'évaluer la qualité pour la faire évoluer dans le temps, pour mettre en place un référentiel, une méthodologie et des normes. La pertinence d'un modèle de qualité est fondée sur sa capacité à modéliser, qualifier, évaluer un certain nombre de règles complexes à partir de mesures (comme par exemple évaluer la structure générale du code source à partir de métriques de code). Il faut également distinguer ce qui donnera lieu à une évaluation de la qualité. En la matière on distingue principalement deux axes : d'une part l'évaluation des processus métier et d'autre part l'évaluation du système d'information conçu en suivant ces processus. Le premier axe est mesuré par des normes telles que CMMI ou la norme ISO 9000 tandis que le second, dans lequel nous plaçons notre étude, est mesuré via des normes telles que ISO 9126 ou SQuaRE. Nous nous intéressons donc uniquement à l'évaluation de la qualité d'un SI (système d'information) et non aux processus métiers sous-jacents qui sont un autre domaine de recherche concomitant. Pour construire notre modèle de qualité, nous nous sommes tout d'abord employé à définir de manière concrète le terme de qualité, puis nous avons cherché à valider une architecture à partir des différents modèles existants et nous avons défini un certain nombre de principes que nous voulons voir respecter par le modèle. Ces principes reposent à la fois sur les différentes recherches que nous avons effectuées mais également sur les exigences et les contraintes qui nous ont été données par les entreprises partenaires de notre recherche. Pour répondre aux attentes des entreprises avec lesquelles nous avons travaillé, nous avons envisagé l'évaluation de la qualité en réunissant deux approches : construire un modèle qui réponde à la fois aux attentes des *managers* mais également utilisable par les différents techniciens. Pour faire la jonction entre ces deux visions, les mesures utilisées doivent être agrégées de manière à faire passer le message délivré au plus bas niveau vers le plus haut niveau : il s'agit de transcrire les informations brutes en informations globales sans pour autant perdre les détails. De plus, nous distinguons principalement deux types de mesures pour évaluer la qualité : les mesures issues de métriques et les mesures reposant sur l'analyse de documents par des experts. En effet, s'il existe de nombreuses métriques permettant l'analyse du code source, il n'est pas possible d'utiliser des métriques pour analyser la qualité d'un cahier des charges ou encore celle de la documentation. Nous avons donc cherché comment exprimer le résultat de toutes ces mesures de manière homogène et uniforme. Etant donné que notre modèle a pour but de devenir une aide à l'amélioration de la qualité, nous avons cherché des méthodes d'aggrégation qui mettent en avant les points faibles détectés et permette d'indiquer les progrès ou les détériorations de la qualité tout au long du cycle de vie du logiciel le plus finement possible. Le modèle Squash repose donc sur un ensemble de propriétés que doit satisfaire le système de notation de la qualité. Les notes résultant de mesures issues de métriques ont fait l'objet d'un premier projet de recherche [3]. Le second projet de recherche a eu pour objectif de qualifier en terme de qualité la validation fonctionnelle d'un logiciel [4]. Le reste de cet article se décompose comme suit. La section 2 présente les différents modèles existants et détaille leurs lacunes. La section 3 présente les propriétés que nous avons définies pour notre modèle de qualité tandis que nous présentons comment la qualité est évaluée dans la section 4. Enfin, nous concluons et nous présentons les futurs travaux que nous réaliserons autour du modèle.

3. http://www.squale.org
4. http://www.squashtest.org

2 Les modèles de qualité existants

Les principaux modèles de qualité existants actuellement sont des modèles hiérarchiques, inspirés du modèle de McCall, qui recensent les principes de qualité en partant des exigences globales et des principes les plus généraux pour descendre vers les métriques qui permettent de les mesurer. Nous présentons les principaux modèles de qualité sur lesquels reposent l'évaluation des SI actuels, quels que soient leur domaine d'application (web, embarqué, temps réel, gestion, etc.).

2.1 Le modèle de McCall : Facteurs Critères Métriques — FCM

McCall et al. (1976) crée en 1977 un modèle pour mesurer le niveau de qualité d'un logiciel pour l'US Air Force. Son modèle est composé de trois couches : les métriques, les critères et les facteurs. Les facteurs représentent la vision externe de la qualité, celle de l'utilisateur. Les facteurs sont assimilés aux caractéristiques du logiciel. Chaque facteur est divisé en critères qui représentent la vision interne de la qualité, celle du développeur. McCall définit vingt-trois critères évalués à partir de métriques. Il définit, pour chaque critère, une liste de propriétés à mesurer. Chaque propriété obtient une valeur et la valeur du critère correspond à la moyenne des valeurs de chaque propriété. Ce modèle est extrêmement complet mais également très difficile à appliquer du fait des 300 métriques qui le composent. Bien qu'implémenté dans plusieurs outils commerciaux, la correspondance entre les métriques et les critères n'est pas clairement définie comme le remarquent Marinescu et Raţiu (2004). Ce modèle présente également une faiblesse importante : le manque de lisibilité. En effet, lorsqu'un critère obtient une faible note, il est difficile, voire impossible, de relier cette note directement au problème qu'elle pointe, surtout lorsque le critère est composé de plusieurs métriques. Il devient alors difficile de comment remédier au problème existant.

2.2 Les Normes ISO 9126 et SQuaRE, qualité des produits logiciels

ISO 9126 est une norme standard internationale ISO/IEC (2001, 2003) visant à évaluer la qualité logicielle. Elle normalise et classifie un certain nombre de principes qualité. La norme ISO 9126 a défini un modèle hiérarchique qui répartit les attributs de qualité en six caractéristiques générales qui définissent la qualité globale d'une application. Une caractéristique spécifie une exigence qualité, fonctionnelle ou non, des clients et des utilisateurs. Chaque caractéristique est décomposée en sous-caractéristiques. Pour chacune de ces sous-caractéristiques la norme propose une série de mesures visant à l'évaluation de la conformité du produit par rapport aux exigences formulées. Le modèle ISO 9126 est désormais remplacé par la norme SQuaRE depuis 2005 (ISO/IEC (2011)). Elle veut répondre aux différents besoins de la qualité selon les acteurs (développeurs, testeurs, utilisateurs, clients). De plus, elle unifie les différents documents normatifs autour de la qualité et définit un modèle d'évaluation de la qualité composé de huit caractéristiques redéfinies de manière beaucoup plus judicieuse, décomposées en sous caractéristiques qui rendent le modèle plus pertinent. Ces normes ne reposent pas uniquement sur les métriques et de nombreuses mesures doivent être manuelles. De plus, ces normes lient les qualités internes et externes d'un logiciel, indiquent un lien de causalité entre ces deux types de caractéristiques mais ne donnent aucune indication quant aux bonnes ou mauvaises valeurs. Elles constituent une bonne approche pour déterminer la qualité d'un logiciel dans

son ensemble et fournir une vue globale satisfaisante. Cependant, elles ne précisent pas de manière explicite comment mesurer les caractéristiques qualité définies et comment les relier aux métriques de bas niveau. Il n'y a aucun *continuum* entre ces deux niveaux. De plus, le fait de devoir adapter le modèle à chaque cas de figure sans avoir de guide précis augmente d'autant sa difficulté de mise en œuvre.

2.3 Le modèle GQM (*Goal Question Metrics*)

Il s'agit d'une approche de la qualité logicielle promue par Basili et al. (1994) qui définit la mesure de la qualité sur trois niveaux : le niveau conceptuel (the *goal* level), le niveau opérationnel (the *question* level), et le niveau quantitatif (the *metric* level). Il s'agit, à partir d'un processus défini en trois points (*Goal Question Metrics*), de définir un plan de qualité correspondant exactement au logiciel évalué. Le niveau conceptuel fixe les objectifs de mesure, le niveau à atteindre en terme de qualité, les objectifs visés par l'entreprise. Il s'agit du but à atteindre : *the Goal*. Le niveau opérationnel définit pour sa part les questions à poser pour déterminer si les objectifs définis au niveau précédent sont atteints : *the Questions*. Le niveau quantitatif détermine quelles métriques doivent être utilisées pour mesurer le niveau précédent. Cette dernière étape passe par le calcul des métriques qui répondent aux questions de l'étape précédente : *the Metrics*. D'après Fenton et Neil (1999), l'idée de Basili et Rombach était d'emprunter des idées simples et générales correspondant aux exigences des démarches qualité dans le but de définir un schéma permettant de déterminer les métriques utiles et adéquates. En effet, d'après Hall et Fenton (1997), un programme de mesures établi sans objectifs ni buts clairs et précis est presque forcément voué à l'échec. Même si cette démarche d'évaluation de la qualité est largement diffusée dans l'industrie comme moyen de déterminer le niveau de qualité d'un système, elle n'est cependant pas sans faille, comme le soulignent par exemple Bache et Neil (1995) : partir du niveau le plus haut a souvent pour contrepartie d'ignorer totalement ce qui est réellement mesurable au niveau le plus bas. Le compromis idéal serait de coupler les programmes de métriques existants avec ce modèle, de sorte que seules les métriques adéquates soient conservées. De plus, cette démarche n'explicite pas clairement comment intégrer les stratégies et les buts spécifiques aux entreprises dans le modèle. Enfin, elle ne sépare pas clairement les différents points de vue entre les managers et les développeurs et ne permet donc pas une lecture claire systématique et aisée des résultats obtenus.

3 Squah, un modèle hiérarchique à quatre couches

Les modèles de qualité déterminent les principes de qualité qu'ils évaluent ensuite selon un système de notation déterminé. Dans cette section, après avoir défini la qualité d'un logiciel, nous présentons l'architecture de Squash et nous détaillons ses propriétés.

3.1 Que signifie évaluer la qualité

Prenons tout d'abord différentes définitions de la qualité :

1. pour la norme ISO 9001 il s'agit du *degré* auquel un ensemble de caractéristiques *remplit les exigences* (du client) (ISO/IEC (2008)) ;

2. le livre *swebok* de Abran et al. (2004) la définit comme un ensemble de *règles* et de *principes* à suivre au cours du développement d'une application afin de concevoir un logiciel répondant aux attentes (du client), le tout sans défaut d'exécution ;
3. la norme ISO 8402 définit la qualité logicielle comme étant la capacité à satisfaire les besoins exprimés et implicites (ISO/IEC (1994)) ;
4. la norme ISO 9126 fait référence à l'objectif à atteindre pour obtenir la qualité nécessaire et suffisante pour répondre aux besoins réels des utilisateurs (ISO/IEC (2001)).

A partir de ces différentes considérations, nous posons pour notre part les définitions et les principes suivants, sur lesquels nous appuyons ensuite nos travaux : (I) la qualité d'un logiciel est le degré auquel un logiciel remplit les besoins et les exigences des clients sans défaut d'exécution ; (II) la qualité logicielle s'exprime sous forme de règles et de principes à suivre pour développer un logiciel capable de répondre aux attentes du client, c'est-à-dire développer un logiciel de qualité ; (III) les besoins d'un client doivent être exprimés avec soin, de manière précise et détaillée et doivent faire l'objet d'une étude minutieuse ; (IV) la **qualité d'un logiciel** repose donc sur deux notions essentielles qui sont **les attentes du client** et **l'absence de défaut**. La notion (I) fait référence à l'objectif fixé en début de conception d'une application : déterminer précisément ce que doit faire le logiciel en fonction des exigences du client. La notion (II) fait référence au savoir-faire des concepteurs du logiciel : comment concevoir un outil fiable et efficace. Déterminer la qualité d'un logiciel consiste donc à mesurer deux éléments complémentaires : d'une part l'adéquation du logiciel avec les objectifs de départ et d'autre part le respect de procédures éprouvées, de standards et de règles de programmation permettant de fournir un logiciel le plus exempt possible de défaut. La qualité d'un logiciel est donc une donnée relative aux attentes et aux objectifs et non une mesure absolue. **Evaluer la qualité** signifie appliquer un ensemble de règles et de mesures afin de calculer la différence entre objectifs attendus et réalisation obtenue. Il faut déterminer le *pourquoi* — ce que doit faire une application, **le point de vue fonctionnel** — et le *comment* — comment atteindre l'objectif, **le point de vue technique**. Un modèle formalise donc des règles décrivant ce que doit être la qualité et y associe des métriques permettant l'évaluation du logiciel en fonction de ces règles.

3.2 L'architecture de Squash

L'architecture de Squash repose sur celle des modèles de qualité existants qui comportent souvent plusieurs niveaux de lecture permettant de classifier les règles en différents domaines (les facteurs) pour obtenir une vue globale et claire de la qualité. Chaque facteur autrement appelé *caractéristique* décrit la qualité d'un domaine, l'ensemble des facteurs représente la qualité dans son intégralité. Les *sous-caractéristiques* autrement appelées *critères* définissent des règles et des principes qui précisent les *facteurs* auxquels ils sont rattachés. Ce niveau détaille le précédent et permet aux *managers* d'affiner l'évaluation qualitative du modèle. Les *métriques* qui définissent le support dans lequel les niveaux supérieurs puisent pour donner une évaluation qualitative des règles et principes de qualité. Ce niveau se fonde sur des métriques et des résultats souvent difficilement interprétables par des non-spécialistes du domaine. Définir des domaines permet de délimiter précisément le champ d'investigation du modèle. Détailler les domaines en principes permet également de déterminer quelles sont les notions à prendre en considération pour son évaluation. Cependant, entre ces définitions et les métriques utilisées pour les évaluer, il existe un manque : les métriques utilisées ne fournissent pas directement

une évaluation de la qualité mais une donnée brute qui doit être interprétée. Que veut dire un nombre de lignes de code égal à 70 par exemple ? Il faut traduire cette donnée brute en terme de qualité pour lui donner du sens (par exemple une méthode de 70 lignes de code est beaucoup trop longue ou acceptable selon les exigences). Les métriques et les mesures utilisées pour noter un critère de qualité servent à évaluer un ensemble de règles techniques de base qui sont sous-entendues dans la définition du critère. Ce qui implique que pour évaluer un critère il faut parfois s'appuyer sur plusieurs mesures. Il s'agit là d'une des principales raisons pour laquelle un modèle hiérarchique en trois couches perd les informations contenues dans les mesures : elles ne sont plus accessibles au niveau des critères. Le modèle Squash a donc introduit un niveau entre les critères et les mesures : les pratiques. Ce niveau permet de décrire précisément les règles techniques qui composent un critère, de préciser les mesures utilisées pour son évaluation mais également de préciser comment les mesures doivent être interprétées en terme de qualité par rapport à la règle technique décrite. Ce niveau constitue le pivot du modèle dans lequel les mesures brutes sont transformées en note de la qualité d'une règle technique de base.

Le modèle retenu Nous avons donc défini de manière formelle un modèle hiérarchique composé de quatre couches, réparties en deux niveaux :

- **le niveau conceptuel** décrit des principes généraux, il est indépendant des exigences de l'entreprise ou des particularités du logiciel. Il est composé de :

  Facteurs. Un facteur définit un domaine d'évaluation de la qualité, un périmètre précis de ce que le modèle analyse. Un facteur regroupe les critères qui constituent les principes conceptuels qui définissent le domaine évalué par le facteur. L'ensemble des facteurs donne une vue globale de la qualité du logiciel. La définition des facteurs d'un modèle de qualité permet donc de préciser les contours d'évaluation de la qualité.

  Critères. Un critère est un principe conceptuel de qualité. Il regroupe les pratiques qui lui sont associées et qui décrivent en règles techniques le principe énoncé par le critère.

- **le niveau technique** est étroitement lié aux particularités du logiciel — par exemple le langage de programmation — et aux exigences de l'entreprise — par exemple le type de tests que l'entreprise effectue sur le logiciel. Il est composé de :

  Pratiques. Une pratique exprime une règle technique de base. Les bonnes pratiques sont des règles techniques à suivre tandis que les mauvaises pratiques sont des principes à éviter. L'ensemble des pratiques d'un domaine représente toutes les règles à appliquer pour obtenir une qualité optimum dans ce domaine ;

  Mesures. Une mesure est une donnée brute qui permet d'évaluer un principe de qualité. Ces mesures peuvent être issues de métriques, de règles, ou de documents attachés au projet mesuré. A un nom de métrique correspond souvent plusieurs façons de mesurer. Il convient donc de reposer l'évaluation de la qualité sur des métriques et des mesures préalablement définies afin de ne pas fausser les résultats.

Les notes du modèle Le système de notation du modèle Squash repose sur l'exigence suivante : noter la qualité de manière homogène et facilement compréhensible par tous les acteurs

de la conception logicielle. Les notes sont calculées dans l'intervalle $[0;3]$ suivant les mêmes significations :

- entre 0 et 1, le but n'est pas atteint ;
- entre 1 et 2, le minimum est atteint mais avec des réserves ;
- entre 2 et 3, le but est atteint.

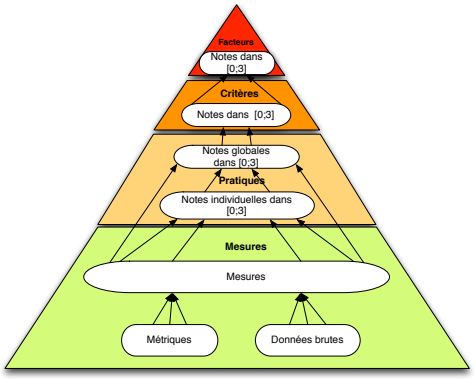

FIG. 1 – *Représentation schématique de Squash*

Comme illustré dans la figure 1, les trois couches supérieures sont notées dans l'intervalle $[0;3]$ tandis que la couche sur laquelle repose les notes — les mesures — est notée selon les intervalles de mesure propres à celles-ci. Cette couche distingue deux types principaux de données : les mesures issues de métriques et les données brutes issues de l'évaluation humaine. Pour cette raison, on distingue deux principaux types de pratiques : les pratiques issues des métriques et les pratiques documentaires reposant sur une évaluation faite par un expert.

3.3 Les propriétés du modèle de qualité

Nous avons recensé les exigences auxquelles notre modèle doit répondre. Ces exigences reposent à la fois sur nos recherches mais également sur les exigences des partenaires industriels de notre projet de recherche.

- **Le modèle doit être exhaustif**. Il doit aborder la qualité selon les deux angles de vue identifiés (fonctionnel et technique) à partir des préconisations du *swebok* (Abran et al. (2004)) ;
- **Il possède une partie fonctionnelle**. La qualité repose sur la satisfaction des besoins. Les attentes du client doivent donc être considérées de deux manières différentes. Il faut, d'une part, qualifier l'expression des besoins et d'autre part, déterminer si le logiciel répond aux besoins, comme le préconise la norme Iso 8402 (ISO/IEC (1994)) ;
- **Il possède une partie technique**. Notre définition de qualité précise que le logiciel doit être sans défaut d'exécution. Il est rare de pouvoir atteindre un tel but. Cependant, pour s'en approcher, il faut que l'application soit développée en respectant un savoir-faire et des processus de conception logicielle reconnus et validés. Evaluer la partie technique d'une application consiste donc à formaliser précisément ce savoir-faire et ces

processus et à vérifier que le développement logiciel a suivi ces principes qui permettront également que le logiciel puisse évoluer dans le temps, que ce soit pour intégrer des corrections (la maintenance logicielle) ou pour intégrer de nouvelles fonctionnalités (l'évolution logicielle). Les règles définies en terme de qualité technique devront donc également tenir compte de ces deux exigences tout comme la NASA le préconise (Esmond B. Marvray (2006)) ;

- **Il possède des règles d'évaluation précises**. Il doit préciser comment les mesures doivent être prises en compte pour évaluer la qualité. Il s'agit ici de déterminer quelles mesures pour quel *critère* mais également quel résultat pour quelle note, tout comme l'indique Plante (1994) ;
- **Il est adaptable**. Les règles définies ainsi que l'évaluation de ces règles doivent prendre en compte les particularités de l'application évaluée. Il s'agit de définir le savoir-faire technique et les exigences de l'entreprise qui accompagnent le type de logiciel évalué dans le but d'apporter une aide pertinente à l'amélioration de la qualité comme voulu par nos partenaires ;
- **Il reflète objectivement un niveau de qualité**. Bien que prenant en compte les particularités du logiciel analysé, les règles définies doivent également être indépendantes du logiciel évalué. En effet, un modèle doit être tout à la fois paramétrable mais également témoin du niveau de qualité. Il faut donc constituer un modèle possédant des niveaux de qualité par défaut qui puissent être adaptables de manière raisonnée, définir des règles générales, adapter le niveau d'exigences mais également fixer des seuils et des limites qui reflètent le plus objectivement possible la qualité. Il s'agit donc de fixer les exigences comme le préconise la norme SQuaRE (ISO/IEC (2011)) ;
- **Il n'est pas un but en soit mais une aide**. Ceci rejoint le commentaire de Wiegers (1996) qui nous alerte sur le fait que l'utilisation de mesures pour motiver plutôt que comprendre, est un piège commun : une mesure n'est intrinsèquement ni vertueuse ni mauvaise, tout simplement informative. Utiliser les mesures pour motiver plutôt que pour apprendre peut potentiellement conduire à un comportement dysfonctionnel, dans lequel les résultats obtenus ne sont pas compatibles avec les objectifs visés par la motivation. Toutefois, en pratique, et dans toute activité humaine, il est difficile de concevoir un modèle de qualité qui n'aura pas tendance à devenir un objectif lui-même. Pour être accepté dans la pratique, un modèle de qualité ne doit pas être uniquement un modèle d'évaluation, mais aussi un guide pour améliorer la qualité. Un *manager* doit savoir si le projet a des problèmes de qualité, tandis que le développeur doit savoir quel composant doit être corrigé. Cela implique que l'évaluation permette également une analyse fine des résultats ;
- **Il reflète finement les efforts de qualité**. Une évaluation qualitative ne doit pas être vue comme une sanction mais comme un outil d'aide à l'amélioration. Il faut donc encourager les efforts et les transcrire finement pour encourager les techniciens à utiliser cet outil. Cette propriété permet comme nous avons pu le constater, à travers l'usage de Squash dans les entreprises, de faire accepter plus facilement le modèle au sein d'une équipe. Ceci permet de définir l'outil comme une aide et non comme une sanction ;
- **Il doit diriger les efforts**. Le modèle doit fournir une aide à l'amélioration de la qualité en mettant en avant les points faibles. Il doit pointer les échecs dans le but de déterminer l'orientation des efforts à mener pour augmenter la qualité, que ce soit au niveau du

logiciel lui-même ou au niveau des processus de conception. Il s'agit là encore d'une demande très forte de la part des entreprises, utilisatrices de notre modèle ;
- **Il possède différents niveaux de lecture**. Il doit s'adresser à tous, techniciens comme décideurs, être une source d'information exploitable utilement dans leur métier. Le décideur doit pouvoir en tirer une vue de la qualité du logiciel mais également une vue des performances de ses équipes. Il doit pouvoir l'utiliser comme support d'amélioration, d'harmonisation des processus mis en place. Un technicien quant à lui doit pouvoir naviguer dans le modèle pour déterminer facilement les problèmes à résoudre ou les points à examiner attentivement, par exemple déterminer le module du code source qui doit être testé plus particulièrement ;
- **Il possède des résultats facilement interprétables**. Les résultats fournis devront être homogènes et offrir une grille d'interprétation simple comme le conseille la norme ISO 9126 (ISO/IEC (2001)).

Une fois l'architecture et les propriétés du modèle définies, nous détaillons à présent comment le modèle évalue la qualité à partir de mesures.

4 L'évaluation de la qualité

Comme le souligne Rosenberg (1998), quand les métriques sont utilisées pour évaluer un projet, il n'y a aucun guide permettant d'interpréter leurs résultats. Le plus souvent l'interprétation des résultats repose sur le sens commun et l'expérience. Déterminer ce qui est acceptable dépend donc de l'expérience des techniciens et des exigences de l'entreprise. Par exemple, certaines entreprises exigent que la *profondeur d'héritage* n'excède pas un certain seuil tandis que d'autres préfèrent se préoccuper davantage de l'architecture générale du code ou encore de l'application des normes de codage. Par conséquent, nous soulignons que pour évaluer la qualité de manière adéquate il convient de prendre en compte l'organisation, les spécificités et les exigences des entreprises.

Exigences pour l'évaluation de la qualité logicielle Tout comme nous avons établi une liste de propriétés à respecter pour le modèle Squash, nous avons également établi une liste d'exigences qui doivent être satisfaites pour parvenir à agréger des mesures avec succès. Ces besoins sont issus à la fois de nos recherches mais également de l'expérience industrielle de nos partenaires lors de la construction de notre modèle. Ces exigences sont classées selon trois critères **doit**, **devrait** et **pourrait** afin de les classer selon leur niveau d'importance (cf. Stapleton (1997)). Les exigences **doit** sont imposées par notre perception de la façon dont les mesures doivent être utilisées, selon deux étapes, la combinaison et l'agrégation ; les exigences **devrait** et **pourrait** reposent sur les propriétés de techniques d'agrégation dans la littérature et notre expérience avec l'aide du modèle Squash dans l'industrie. La liste de ces exigences est détaillée dans Mordal-Manet et al. (2011).

Les notes issues de métriques Les pratiques de métriques reposent sur un système de notation en deux étapes : la première, appelée combinaison, consiste à grouper plusieurs métriques pour évaluer une pratique. Il s'agit d'une opération sémantique. Cette étape permet également de transformer les résultats de métriques dans l'intervalle $[0; 3]$. Chaque pratique obtient donc

une note par composant, appelée note individuelle. La seconde étape, appelée agrégation, permet de regrouper tous les résultats pour donner une note globale à la pratique dans l'intervalle $[0; 3]$. Il s'agit d'une opération de type statistique effectuée à l'aide d'une fonction continue pondérée pour calculer la note globale du projet pour une pratique donnée. Mordal et al. (2012) détaille et valide ce calcul.

4.1 Les notes issues de mesures non issues de métriques : de l'usage de la logique floue

Certaines pratiques reposent sur des avis humains qu'il est difficile d'interpréter directement sous forme d'une évaluation numérique précise. En nous inspirant de la méthode d'évaluation du modèle McCall, nous avons tout d'abord établi *une liste de caractéristiques* à évaluer pour chacune des pratiques documentaires. Puis, nous avons cherché comment évaluer cette liste, en utilisant une échelle de termes linguistiques *ad hoc* pour recueillir les avis des experts sur ces listes. Une fois cette échelle linguistique fixée nous avons cherché à transposer les avis des experts en une note comprise dans l'intervalle continu $[0; 3]$. Les approches linguistiques fondées sur les sous-ensembles flous ont prouvé leur efficacité pour la modélisation de l'information qualitative. Cette approche consiste à estimer des valeurs linguistiques à l'aide de variables linguistiques dont les valeurs ne sont pas des nombres mais des mots issus d'un langage artificiel ou naturel Zadeh (1975). Les valeurs linguistiques sont modélisables grâce à des sous-ensembles flous Zadeh (1965) qui permettent de capturer les imprécisions du langage. Les paliers obtenus en notation discrète sont ainsi lissés et amènent à une expression plus fine des notes.

Le modèle de Herrera & Martínez Parmi les modèles existants pour représenter l'information imprécise, nous avons retenu les *2-tuple linguistiques* de Herrera et Martínez (2001); Herrera et al. (2008) : ils utilisent des sous-ensembles flous très simples combinés à une valeur de "décalage". Dans ce modèle, les fonctions d'appartenance (les sous-ensembles flous) triangulaires sont suffisantes pour capturer l'imprécision des propositions, ce qui permet de ne pas avoir à utiliser un modèle de logique flou trop complexe par rapport à nos besoins tout en obtenant une note dans un intervalle continu, grâce aux valeurs de décalage du modèle. De plus ils mettent en oeuvre des descripteurs linguistiques précis pour former l'ensemble de termes attaché à chaque univers de discours. L'utilisation de ce modèle pour le calcul des notes dans le modèle Squash a fait l'objet d'une explication détaillée dans Mordal. Etant donné un terme linguistique, le formalisme 2-tuple fournit un couple (sous-ensemble flou, translation symbolique) $= (s_i, \alpha)$ avec $\alpha \in [-0.5, 0.5[$. α permet de supporter les calculs sans perte d'information, lorsque le résultat ne coïncide pas exactement avec l'un des sous-ensembles de départ, ce qui a motivé notre choix pour le calcul des notes des pratiques documentaires du modèle Squash. De plus, comme expliqué dans Mordal, nous avons choisi une information linguistique multigranulaire traduits en partition floue comme expliqué par Herrera et al. (2008). La figure 2 détaille la traduction sous forme de 2-tuple des notes attribuées dans l'échelle de valeurs non uniformément réparties de la figure 3 et reprise en bas de la figure 2. Ce modèle de logique floue permet donc de représenter de manière pas obligatoirement uniformément distribuée des informations linguistiques ce qui correspond exactement à ce que nous souhaitons pour l'évaluation des pratiques documentaires, l'échelle de valeurs retenue étant non unifor-

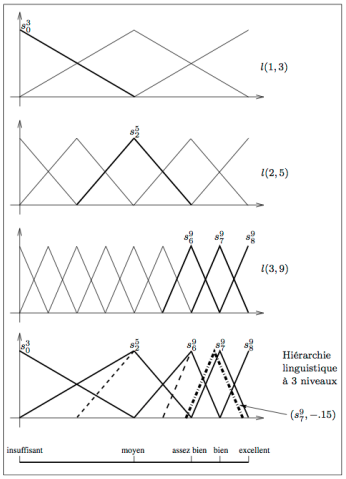

FIG. 2 – *Exemple de partition floue utilisant une hiérarchie linguistique à 3 niveaux.*

mément distribuée. De plus il permet d'effectuer des calculs tels que des agrégations de valeurs tout en conservant précisément les décalages obtenus (grâce à α, le second terme du 2-tuple). Nous avons donc utilisé leur approche pour l'évaluation des pratiques documentaires.

L'évaluation des caractéristiques de Squash Squash utilise les termes linguistiques $[$*excellent*, *bien*, *assez bien*, *moyen*, *insuffisant*$]$ répartis selon l'échelle de la figure 3. Ces termes permettront à l'expert d'évaluer les caractéristiques d'une pratique documentaire donnée. Cette représentation traduit les exigences attendues en terme de qualité dans le modèle Squash : les caractéristiques qui n'atteignent pas la moyenne sont recalées tandis que celles au dessus de la moyenne sont classées plus finement pour déterminer de manière plus précise leur niveau de qualité. L'idée est d'utiliser une échelle qui soit symboliquement similaire aux formules utilisées pour calculer les notes individuelles des pratiques de code. En effet, ces formules ont été choisies pour faire chuter les notes en dessous de un rapidement, et pour affiner la note obtenue dans l'intervalle $[1; 3]$. Ce même principe sous-tend le choix de l'échelle. Chaque caractéris-

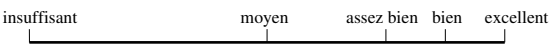

FIG. 3 – *Notes non uniformément distribuées sur l'axe.*

tique est donc représentée selon le modèle détaillé dans le tableau 1. La première ligne du second tableau donne les sous-ensembles flous associés aux termes linguistiques selon l'échelle choisie (cf. Figure 3) et selon le modèle présenté dans la figure 2 ; la seconde ligne donne les sous-ensembles flous ramenés dans une échelle commune (cette échelle commune nous permet ensuite d'agréger les différents termes). Le premier tableau donne les sous-ensembles associés aux caractéristiques de type binaire : *oui*, *non*. En effet, les caractéristiques de type binaire sont

Terme linguistique	oui	non
Caractéristique de type binaire	s_1^2	s_0^2

Terme linguistique	excellent	bien	assez bien	moyen	insuffisant
Caractéristique non binaire dans la hiérarchie à 3 niveaux	s_8^9	s_7^9	s_6^9	s_2^5	s_0^3
Caractéristique non binaire exprimée dans la hiérarchie commune	s_8^9	s_7^9	s_6^9	s_4^9	s_0^9

TAB. 1 – *Modèle de représentation d'une caractéristique et les 2-tuple associés.*

modélisées pour les calculs comme des 2-tuple dans une hiérarchie à 2 étiquettes seulement : s_0^2 et s_1^2, sachant que s_0^2 correspond à *non* et s_1^2 correspond à *oui*. Cette façon de représenter les informations des pratiques documentaires nous permet également une évaluation individuelle précise de chacune des caractéristiques : en cas de mauvaise note attribuée à une pratique, le détail des caractéristiques nous renseigne directement sur ce qu'il faut améliorer pour parvenir à augmenter le niveau de qualité. Cette évaluation individuelle joue ici le même rôle que les notes individuelles des pratiques de code. Prenons l'exemple de la pratique *qualité de la documentation*. Celle-ci est évaluée avec une liste de caractéristiques associée à des sous-ensembles flous selon le modèle représenté dans le tableau 2, qui différencie les caractéristiques de type binaire — *oui*, *non* — aux caractéristiques plus complexes auxquelles sont associées cette fois les termes linguistiques [*excellent*, *bien*, *assez bien*, *moyen*, *insuffisant*]. Les caractéristiques de type binaire telles que C_1 et C_2 sont modélisées uniquement pour les besoins des calculs dans une hiérarchie à deux étiquettes. Les autres caractéristiques sont modélisées dans une hiérarchie à trois niveaux telle que décrite précédemment (voir les figures 2 et 3). Les caractéristiques une fois définies avec leurs différentes valeurs, nous nous sommes alors attelés à agréger celles-ci pour parvenir à évaluer la pratique. Selon Truck (2002) il s'agit ici de savoir comment exprimer le résultat d'un processus cognitif ou encore exprimer le résumé d'un ensemble de données quelconques par une réponse agrégée, représentative ou encore émergente.

Les opérateurs OWA Il faut remarquer que notre système utilise 3 hiérarchies différentes réparties comme telles : les valeurs comprises entre s_0^3 et s_2^5 exclus (donc depuis s_0^3 jusqu'aux valeurs "à gauche" de s_2^5) sont dans une hiérarchie à 3 étiquettes, les valeurs comprises entre s_2^5 et s_6^9 exclus sont dans une hiérarchie à 5 étiquettes et les valeurs supérieures sont dans une hiérarchie à 9 étiquettes, comme expliqué précédemment. Ainsi, pour agréger les valeurs des évaluations, nous devons tout d'abord passer dans une hiérarchie commune à 9 étiquettes et exprimer le résultat dans cette hiérarchie. Dans cette dernière, les différents sous-ensembles deviennent donc (voir la figure 2) :

Nom	Caractéristique	oui	non
C1	Existence de la documentation technique	s_1^2	s_0^2
C2	Existence des normes et des règles de mise à jour	s_1^2	s_0^2

Nom	Caractéristique	excellent	bien	assez bien	moyen	insuffisant
C3	La documentation est complète	s_8^9	s_7^9	s_6^9	s_4^9	s_0^9
C4	Mise à jour de la documentation	s_8^9	s_7^9	s_6^9	s_4^9	s_0^9
C5	Suivi des règles	s_8^9	s_7^9	s_6^9	s_4^9	s_0^9
C6	Pertinence de la documentation	s_8^9	s_7^9	s_6^9	s_4^9	s_0^9
C7	Remplissage des *templates*	s_8^9	s_7^9	s_6^9	s_4^9	s_0^9

TAB. 2 – *Liste des caractéristiques et de leur 2-tuple associé pour la pratique* qualité de la documentation.

- s_0^3 devient s_0^9. En effet, le sous-ensemble s_0^3 exprimé dans une hiérarchie à 9 étiquettes correspond à s_0^9 ;
- s_2^5 devient s_4^9 ;
- les trois sous-ensembles suivants sont déjà dans la hiérarchie souhaitée.

Le résultat obtenu devra ensuite être repassé dans la hiérarchie multiple en fonction de la valeur obtenue. Nous avons retenu les *OWA* pour *Ordered Weighted Averaging operators*. Cette famille d'opérateurs a été introduite par Yager (1988) pour fournir un moyen d'agréger des objets en satisfaisant plusieurs critères. Les *OWA* sont à la fois conjonctifs et disjonctifs : on peut agréger des objets en établissant que "tous les critères" ou "certains critères" ou "au moins un critère", ou etc. "doivent être satisfaits", tout en ayant des pondérations sur chaque critère. Les *OWA* se définissent de la sorte : Un opérateur d'agrégation $\mathcal{A}$, $\mathcal{A} : [0,1]^n \rightarrow [0,1]$, est appelé un OWA de dimension n s'il est associé à un vecteur de pondération W :

$$\begin{bmatrix} w_1 \\ \dots \\ w_n \end{bmatrix}$$

tel que $w_j \in [0,1], \sum_{i=1}^{n} w_i = 1$ et $\mathcal{A}(x_1, x_2, \dots, x_n) = \sum_{j=1}^{n} w_j y_j$ et où y_j est le j-ième plus grand élément des $\{x_1, x_2, \dots x_n\}$. On remarque que, lorsque $W = (1, 0, 0, \dots, 0)$ (*i.e.* tous les poids sont nuls sauf le premier), l'opérateur est un $\max$. De même, lorsque $W = (0, 0, \dots, 0, 1)$ (*i.e.* tous les poids sont nuls sauf le dernier), l'opérateur est un $\min$. Enfin, lorsque $W = (1/n, 1/n, \dots, 1/n)$ (*i.e.* tous les poids sont égaux), l'opérateur est une moyenne arithmétique.

Nom	Caractéristique	oui	non	Poids
C1	Existence	s_1^2	s_0^2	$0,2$
C2	Normes	s_1^2	s_0^2	$0,1$

Nom	Caractéristique	excellent	bien	assez bien	moyen	insuffisant	Poids
C3	Intégralité	s_8^9	s_7^9	s_6^9	s_4^9	s_0^9	$0,2$
C4	Mise à jour	s_8^9	s_7^9	s_6^9	s_4^9	s_0^9	$0,2$
C5	Règles	s_8^9	s_7^9	s_6^9	s_4^9	s_0^9	$0,1$
C6	Pertinence	s_8^9	s_7^9	s_6^9	s_4^9	s_0^9	$0,16$
C7	templates	s_8^9	s_7^9	s_6^9	s_4^9	s_0^9	$0,04$

TAB. 3 – *Liste des caractéristiques avec les valeurs et les poids associés qui qualifient la pratique* qualité de la documentation.

L'agrégation des caractéristiques de Squash Les *OWA* permettent de pondérer une caractéristique en fonction de son importance pour le calcul de la note finale. Nous utilisons donc ce type d'agrégation pour nos pratiques. Par exemple, le tableau 3 regroupe les caractéristiques qui définissent la pratique *qualité de la documentation* avec les valeurs linguistiques et les poids associés à chacune. La somme des poids atteint le total de 1. Pour être au plus près des exigences de notre modèle, à chaque pratique est associé un algorithme de calcul des notes afin de prendre en compte certaines caractéristiques bloquantes dans le calcul de la note. Par exemple, dans ce cas précis, la caractéristique *Existence de la documentation technique* est dite bloquante. En effet, s'il n'existe pas de documentation technique, il s'agit alors d'un défaut majeur et la note de la pratique est immédiatement de 0. Une fois la note agrégée obtenue, nous devons alors la convertir en valeur numérique comprise dans l'intervalle $[0; 3]$ pour être en harmonie avec les notes du modèle selon l'échelle de la figure 4. Squash évalue la qualité.

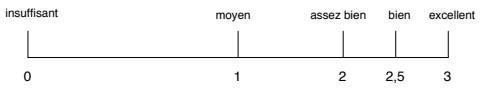

FIG. 4 – *Echelle de conversion numérique de la note*

Selon ce principe, une appréciation moyenne ne peut pas être considérée suffisante d'un point de vue qualité. Il faut être *au dessus de la moyenne* pour obtenir une note qualitative correcte : *avoir atteint le but mais avec des réserves*. Le terme *moyen* prend donc le sens suivant dans le cadre du modèle : une appréciation moyenne correspond à un but atteint mais avec réserves

et non pas à un but suffisant. En rapportant les significations des notes du modèle aux termes linguistiques choisis, les notes deviennent donc :

- les notes autour de *insuffisant* se traduisent par : *presque nul* ;
- les notes autour de *moyen* se traduisent par : *minimum absolument pas atteint* pour les notes inférieures à *moyen* et *minimum atteint avec réserves* pour les notes supérieures à *moyen* ;
- les notes autour de *assez bien* se traduisent par : *minimum atteint avec réserves* pour les notes inférieures à *assez bien* et *but atteint* pour les notes supérieures à *assez bien* ;
- les notes autour de *bien* se traduisent par : *but atteint* pour les notes inférieures à *bien* et *but atteint presque parfaitement* pour les notes supérieures à *bien* ;
- les notes autour de *excellent* se traduisent par *but parfaitement atteint*.

Ceci permet donc d'obtenir une note dans l'intervalle $[0; 3]$ de manière continue et pondérée : d'une part les différentes caractéristiques sont pondérées pour refléter leur importance par rapport à la pratique mais également les unes par rapport aux autres, d'autre part la transposition des termes linguistiques en note se fait de manière continue. D'autre part, ce système de notation des pratiques reposant sur des données brutes peut être étendue à tout autre type de notation. En effet, les 2-tuple peuvent être traduits dans n'importe quelle autre échelle de valeur, la conversion numérique n'étant pas rattachée à ce type de notation préliminaire. Ainsi, il peut donc être envisagé de convertir le 2-tuple final dans un autre système de notation.

5 Conclusion et perspectives

Nous avons présenté le modèle Squash, un modèle homogène d'évaluation de la qualité qui propose principalement deux méthodes d'évaluation selon le type de mesures utilisées. Une première version du modèle a été implémentée sous forme d'un logiciel open-source, principalement consacré à évaluer la qualité à partir de mesures issues de métriques de code. Ce modèle a fait l'objet d'un projet de recherche et est utilisé par les entreprises Air France-KLMet PSA Peugeot-Citroën. Une seconde version a été implémentée sous forme d'un plugin au logiciel Sonar lors d'un second projet de recherche consacré à la qualité de la validation fonctionnelle. Celui-ci est actuellement en cours de validation chez Generali. Le modèle Squash se compose de deux niveaux : le niveau conceptuel qui donne une vue globale de la qualité, principalement destinée aux *managers* et le niveau technique composé de règles décrivant les principes techniques propres au logiciel évalué et destiné principalement aux développeurs et aux testeurs. De plus, le modèle satisfait les exigences définies et évalue la qualité selon un système de notation continue et homogène, facilement interprétable. Les notes sont entièrement décomposables de manière à identifier précisément les composants ou les processus ne respectant pas les exigences de qualité définies par l'entreprise. Il évalue la qualité en respectant les deux grands principes qui le sous-tendent, à savoir, mettre en évidence les points faibles et traduire finement les variations de qualité au fil du temps. De plus, bien que le modèle Squash ait fait l'objet d'une validation empirique auprès des partenaires industriels des deux projets de recherche, il apparaît aujourd'hui indispensable de déterminer un processus de comparaison des différents modèles et des normes existants. En effet, les modèles actuellement proposés ne permettent pas une comparaison directe puisque les domaines d'évaluation et les techniques d'évaluation y attenant ne sont pas totalement identiques. Il convient donc de travailler à mettre en place une méthode objective de comparaison et d'analyse de ces modèles.

Références

Abran, A., P. Bourque, R. Dupuis, et L. Tripp (2004). Guide to the software engineering body of knowledge (ironman version). Technical report, IEEE Computer Society.

Bache et Neil (1995). Introducing metrics into industry : a perspective on gqm. *Software Quality Assurance and metrics : A Worldwide prespective*, 59–68.

Basili, V. R., G. Caldiera, et H. D. Rombach (1994). The goal question metric approach. In *Encyclopedia of Software Engineering*. Wiley.

Esmond B. Marvray, NASA Goddard Space Flight Center, S. Q. (last update 2006). Procedure for developing and implementing software quality programs.

Fenton, N. E. et M. Neil (1999). Software metrics : successes, failures, and new directions. *Journal of Systems and Software*.

Hall, T. et N. Fenton (1997). Implementing effective software metrics programs. *IEEE Software 14*, 55–65.

Herrera, F., E. Herrera-viedma, et L. Martínez (2008). A fuzzy linguistic methodology to deal with unbalanced linguistic term sets. *IEEE Transactions on Fuzzy Systems*, 354–370.

Herrera, F. et L. Martínez (2001). A model based on linguistic 2-tuples for dealing with multigranularity hierarchical linguistic contexts in multiexpert decisionmaking. *IEEE Transactions on Systems, Man and Cybernetics. Part B : Cybernetics*.

ISO/IEC (1994). Iso/iec 8402 management de la qualité et assurance de la qualité.

ISO/IEC (2001). Iso/iec 9126-1 software engineering -product quality- part 1 : Quality model.

ISO/IEC (2003). Iso/iec 9126-3 software engineering -product quality- part 3 : Internal metrics.

ISO/IEC (2008). Iso/iec 9001 management de la qualité.

ISO/IEC (2011). Iso/iec 25010 :2011 software engineering -software product quality requirement and evaluation.

Marinescu, R. et D. Raţiu (2004). Quantifying the quality of object-oriented design : the factor-strategy model. In *Proceedings 11th Working Conference on Reverse Engineering (WCRE'04)*, Los Alamitos CA, pp. 192–201. IEEE Computer Society Press.

McCall, J., P. Richards, et G. Walters (1976). *Factors in Software Quality*. NTIS Springfield.

Mordal, K. Evaluer la qualite d'un système d'information dans son ensemble. In *CAL2013 : conférence francophone sur les architectures logicielles*.

Mordal, K., N. Anquetil, J. Laval, A. Serebrenik, B. Vasilescu, et S. Ducasse (2012). Software quality metrics aggregation in industry. *Journal of Software : Evolution and Process*.

Mordal-Manet, K., J. Laval, S. Ducasse, N. Anquetil, F. coise Balmas, F. Bellingard, L. Bouhier, P. Vaillergues, et T. McCabe (2011). An empirical model for continuous and weighted metric aggregation. In *15th Eur. Conf. Soft. Maintenance and Reeng.*, pp. 141–150. IEEE.

Plante, J. (1994). *Evaluation de programmes*. Presse de l'universit/'e Laval, Qu/'ebec.

Rosenberg, L. H. (Software Technology Conference (Utah - April 1998)). Applying and interpreting object oriented metrics.

Stapleton, J. (1997). *DSDM Dynamic Systems Development Method : The Method in Practice*. Addison-Wesley.

Truck, I. (2002). Approches symbolique et floue des modificateurs linguistiques et leur lien avec l'agrégation. Master's thesis.

Wiegers, K. E. (1996). Software process improvement : Ten traps to avoid. *Software Development 4*, 51–58.

Yager, R. R. (1988). On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Trans. Syst. Man Cybern. 18*(1), 183–190.

Zadeh, L. A. (1965). Fuzzy sets. *Information Control*, 338–353.

Zadeh, L. A. (1975). The concept of a linguistic variable and its application to approximate reasoning - i. *Inf. Sci. 8*(3), 199–249.

Summary

We present a model for analysing and evaluating the quality of industrial software, named Squash model. We determine an evaluation methodology for both technical and functional quality principles based on studying ISO 9126 and SQuaRE standards, industrial skill in software development and functional validation. From our methodology, we formalized a quality model taking into account the particularities and requirements of industrial applications. The implementation of this model is being validated at Generali. This model is divided into two levels: the conceptual quality level that defines the main quality principles and the technical quality that defines basic technical rules and measures used to assess the top level.