École Polytechnique de Montréal

Département de Génie Logiciel

INF8301 – Qualité logicielle

Travail pratique #3 – Une évaluation d'un document d'architecture

1. Objectifs

Les objectifs de ce laboratoire sont :

- De mettre en commun le matériel du cours présenté depuis le début de la session,
- D'effectuer une évaluation de qualité dans un contexte réel,
- De découvrir comment mesurer la qualité d'un plan d'architecture au niveau conceptuel.

2. Travail à effectuer

L'analyse du contexte du client a déjà été faite pour le TP2, donc il n'est pas nécessaire de recommencer. Il faudrait simplement corriger les erreurs trouvées, s'il y a lieu, et remettre cette partie dans le rapport pour le TP3.

Le travail à faire est donc :

- **Déterminer les facteurs de qualité importants pour la discipline et pour le client** : Ce travail pratique évalue le document d'architecture et concerne donc la discipline d'analyse et conception.
- **Développer un modèle de qualité pour le document d'architecture** : À la lumière du contexte du client et du contexte de la discipline, choisissez et définissez un ou plusieurs facteurs de qualité pertinents.
    - Utilisez l'approche GQM pour développer les facteurs de qualité en questions, et finalement en métriques.
- **Décrire les métriques choisies pour évaluer la qualité** : Vous devez présenter au moins huit (8) métriques qui seront évaluées sur le document d'architecture.

    - La section 3 présente quelques recommandations pour le choix des métriques.
    - La définition des métriques doit utiliser le gabarit de l'ISO 25021, comme pour le TP2.
- **Effectuer la mesure du document d'architecture** : Les membres de l'équipe évaluent l'architecture. Idéalement, cette mesure se fait avec quelqu'un qui fait le projet intégrateur et quelqu'un qui ne le fait pas, afin de bénéficier d'un point de vue interne et d'un point de vue externe.
    - Documentez clairement où vous trouvez des problèmes. Il n'est pas suffisant de dire que l'architecture n'est pas bonne. Il faut indiquer les problèmes que vous allez trouver. L'objectif est d'aider l'équipe du projet intégrateur dans leur révision des exigences
- **Présentez vos interprétations des résultats obtenus** : Présentez les problèmes majeurs trouvés dans le document et vos recommandations afin d'éviter ces problèmes dans l'avenir.
    - L'interprétation des mesures doit être faite avec attention. Cette section sera donc importante pour la correction.
- **Temps passé sur le laboratoire** : Nous travaillons constamment à l'amélioration des laboratoires du cours d'INF8301. Si vous pouviez indiquer le temps nécessaire pour réaliser les travaux, cela nous permettrait de réajuster la charge de travail lors des prochaines sessions.

3. Guide pour la conception des métriques

L'évaluation de qualité d'une architecture de niveau conceptuelle n'est pas un exercice facile. Certains aspects mineurs peuvent être évalués quantitativement, mais la plupart des évaluations sont qualitatives, et donc très floues. Il y a cependant des pratiques que nous pouvons respecter afin de limiter la subjectivité au maximum.

3.1. Sources d'inspiration pour les métriques

Les 21 principes de Microsoft et les 8 patrons de Buschmann peuvent vous aider à déterminer des questions de qualité importantes pour l'architecture d'un logiciel. Ces principes et patrons sont utiles pour évaluer la qualité du **contenu** du document d'architecture :

Les 21 principes de Microsoft sont disponibles dans les pages 11 à 14 (*Key Design Principles*) du *Microsoft Application Architecture Guide, 2nd Edition*.

Les 8 patrons de Buschmann sont décrits dans les pages suivantes :

- Layers: http://www.vico.org/pages/PatronsDisseny/Pattern%20Layers/
- Pipes and Filters: http://www.vico.org/pages/PatronsDisseny/Pattern%20Pipes%20and%20Filters/
- Blackboard: http://www.vico.org/pages/PatronsDisseny/Pattern%20Blackboard/
- Model View Controller (MVC) : http://www.vico.org/pages/PatronsDisseny/Pattern%20Model%20View%20Controller/
- Presentation Abstraction Control (PAC) : http://www.vico.org/pages/PatronsDisseny/Pattern%20Presentation%20Abstra/
- Microkernel: http://www.vico.org/pages/PatronsDisseny/Pattern%20MicroKernel/
- Reflection: http://www.vico.org/pages/PatronsDisseny/Pattern%20Reflection/
- Broker: http://www.vico.org/pages/PatronsDisseny/Pattern%20Broker/

Afin de comprendre le lien qui peut être fait entre patrons d'architecture et facteurs de qualité, vous pouvez voir l'article d'Harrison et Avgeriou (2007), qui présente comment chaque patron de Buschmann affecte certains facteurs de qualité de haut niveau.

Notez que je ne crois pas qu'il soit possible de définir des métriques applicables à tous les projets logiciels. L'architecture est un facteur très lié aux particularités du projet, et dépend donc beaucoup des contextes (clients, organisation, projet, équipe). Vos métriques seront probablement uniques au projet que vous étudiez.

3.2. Éviter les métriques trop génériques

Par exemple, prenons une équipe qui veut mesurer la qualité de performance du futur logiciel modélisé dans le document d'architecture. Le gestionnaire demande alors aux développeurs d'évaluer sur une échelle de 1 à 5 la performance du futur système selon l'architecture. Cette approche est trop générique et trop subjective.

L'astuce est de décomposer progressivement le facteur de haut niveau en éléments de plus bas niveau (questions et métriques). Par exemple :

- Facteur de qualité : Performance du futur système
    - Question : L'architecture utilise-t-elle des patrons appropriés pour contrôler la performance dans les potentiels goulots d'étranglement (*bottlenecks*) ?

- Métrique : Liste à cocher (*checklist*) des futurs goulots d'étranglement du système et de la présence d'un patron pour mitiger l'impact de ce goulot sur la performance.

Pensez à une évaluation académique. Si un professeur vous dit que vous avez eu 25% pour un travail, sans expliquer pourquoi, c'est très difficile pour vous de comprendre et de vous améliorer. En décomposant la note en sous-éléments, il est plus facile de comprendre où les erreurs se trouvent.

3.3. Définir clairement les niveaux d'évaluation

Il arrive qu'une évaluation n'ait pas le choix d'être subjective. Dans ce cas, les niveaux d'évaluation doivent être clairement définis, avec des frontières claires. L'évaluation des niveaux doit être le plus claire possible, afin d'assurer que deux personnes faisant la même évaluation mettent le même niveau. Cela permet aussi d'assurer que l'interprétation des différents niveaux se fait adéquatement.

Encore une fois, la décomposition en sous-éléments permet de définir adéquatement les niveaux d'une évaluation subjective. Par exemple, pour évaluer la portabilité d'un module :

- Niveau faible : Certaines parties du module dépendent des particularités de la plateforme visée (ex.: Utilisation de classes partielles en C#, utilisation de librairies seulement disponibles en C++, etc.).
- Niveau moyen : Utilisation de patrons appropriés pour assurer un couplage minimal entre le module et ses modules voisins.
- Niveau élevé : Applique les principes suivants qui permettent de porter des sous-sections du module à d'autres plateformes :
    - Separation of concerns,
    - Single Responsibility principle,
    - Keep crosscutting code abstracted from the application business logic.

De cette manière, il ne s'agit pas seulement d'une évaluation subjective faible/moyen/élevée, mais d'une évaluation ancrée sur des observations vérifiables.

3.4. Assurer de bien couvrir le facteur de qualité

Un facteur de qualité peut être approché de différents points de vue. Quand un facteur de qualité est flou, ces points de vue complémentaires peuvent permettre d'avoir une meilleure vue d'ensemble de ce facteur.

Par exemple, la maintenabilité peut être évaluée des points de vue suivants, selon la norme ISO 25010 (page 12/44 et page 23/44) :

- Modularité : Qualité d'un système qui fait que les changements dans un module a un minimum d'impact sur les autres modules.
- Réutilisabilité : Degré qu'un module peut être utilisé dans plus d'un système.
- Analysabilité : Facilité de comprendre le fonctionnement d'un module, grâce aux artéfacts disponibles.
- Modificabilité : Degré qu'un module peut être modifié sans introduire des défauts ailleurs. Par exemple, la présence de beaucoup d'attributs dans une classe peut diminuer sa modificabilité en rendant les modifications difficiles.
- Testabilité : Difficulté de déterminer des critères de fonctionnement d'un module, et d'évaluer si les critères ont été respectés.

De même, ces sous-facteurs (ou questions selon l'approche GQM) peuvent être approchées selon différents points de vue. Par exemple, la modularité peut être évaluée avec les éléments suivants :

- Point de vue du code : Métriques de cohésion et couplage.
- Point de vue des principes architecturaux : Principes augmentant la cohésion et diminuant le couplage.
- Point de vue des patrons utilisés : Utilisation de patrons améliorant la modularité.

4. Livrable à remettre, procédure de remise et pénalité de retard

Il n'y a qu'un rapport texte à remettre, en format Word, OpenOffice ou PDF, contenant les résultats de votre analyse de qualité. Le résultat de votre travail sera transmis à toute l'équipe faisant le projet intégrateur associé : Le niveau de détail nécessaire doit donc correspondre à leurs besoins.

Seule une remise électronique est exigée, à travers le site Moodle. Si la remise sur Moodle ne fonctionne pas, vous pouvez envoyer le travail par courriel au mathieu.lavallee@polymtl.ca.

La date limite de remise du travail pratique est le vendredi 28 février 2013, à midi, pour les équipes B1 et B2.