Documentation des Décisions d'Architecture

Contexte

L'architecture pour les projets agiles a besoin d'être décrite et définie différemment. Toutes les décisions ne peuvent pas être prises en une seule fois, de la même manière qu'elles ne seront pas toutes prises au démarrage du projet.

Les méthodes agiles ne sont pas opposées à la documentation, seulement à la documentation sans valeur. Les documents qui assistent l'équipe peuvent avoir de la valeur, mais uniquement s'ils sont tenus à jour. Les gros documents ne sont jamais tenus à jour. Les documents petits et modulaires ont au moins une chance d'être tenus à jour.

Personne ne lit non plus les gros documents. Beaucoup de développeurs sont déjà passés sur au moins un projet où le document de spécification était plus gros (en octets) que le taille totale du code source. Ces documents sont trop gros pour être ouverts, lus ou mis à jour. Des documents plus courts sont plus faciles à consommer pour toutes les parties prenantes.

Une des choses les plus difficiles à suivre durant la vie d'un projet est la raison derrière certaines décisions. Un nouvel arrivant sur le projet pourra rester perplexe, déconcerté, enchanté ou exaspéré par certaines de ces décisions passées. Sans comprendre le raisonnement ou les conséquences, cette personne n'a que 2 options :

1. Accepter la décision aveuglément.

Cette réponse peut être ok, si la décision est toujours valide. Cela peut ne pas être bon cependant, si le contexte a changé et que la décision devrait être revisitée. Si le projet accumulent trop de décisions acceptées sans les comprendre, alors l'équipe de développement hésitera à changer quoique cela soit et le projet s'écroulera sous son propre poids.

2. La changer aveuglément.

Encore une fois, cela peut être ok, si la décision doit être inversée. Néanmoins, changer la décision sans comprendre les motivations ou les conséquences qui lui sont associées peut être dommageable au projet dans son ensemble sans même sans rendre compte (par exemple la décision prenait en compte une exigence non-fonctionnelle non encore testée pour l'instant).

Il est préférable d'éviter tant l'acceptation que le changement aveugle.

Décision

Nous allons archiver les décisions d'architectures significatives : celles qui affectent la structure, les caractéristiques non-fonctionnelles, les dépendances, les interfaces ou les construction techniques.

L'Enregistrement d'une Décision d'Architecture (**NDT** : en anglais **ADR** pour *Architectural Decision Record*. On garde cet acronyme dans le reste du texte plutôt que d'utiliser **EDA**) est un court fichier texte dans un format similaire à celui des modèles de conception de Christopher Alexander (bien que les décisions elles-mêmes ne soit pas nécessairement des modèles de conception, elles en partagent l'équilibre caractéristique des forces). Chaque enregistrement décrit un ensemble de forces et de contraintes et une unique décision en réponse à ces forces et

ces contraintes. Il faut noter que la décision est le point central ici, en conséquence des forces et contraintes spécifiques peuvent apparaître dans de multiples **ADR**.

On gardera les ADR dans le dépôt du projet sous `doc/arch/adr-NNN.md`

Il est préférable d'utiliser un language de formattage de texte léger comme Markdown ou Textile.

Les ADR seront numérotés séquentiellement et de façon monotone (**NDT** : i.e que ces nombres ne font que croître ou décroître). Les numéros ne seront pas réutilisés. S'il y a un retour arrière sur une décision, on gardera présent l'ancienne décision mais en la marquant comme *remplacée*. En effet, il est toujours toujours pertinent de savoir que c'était une décision mais que cela n'en est plus une.

On utilisera un format avec seulement quelques parties, de sorte que chaque document soit facile à appréhender. Le format n'a que quelques parties.

Titre

Ces documents ont des noms qui sont de courtes phrases nominales. Par exemple, **ADR 1 : Déploiement sous Ruby On Rails 3.0.10** ou **ADR 9 : LDAP pour de l'Intégration Multitenant**

Contexte

Cette section décrit les forces et les contraintes en jeu, en incluant celles technologiques, politiques, sociales et locales au projet. Ces différentes forces et contraintes sont probablement en tension et doivent apparaître comme telles. Le ton de cette section doit être neutre et ne pas refléter une prise de position. Il s'agit de juste décrire les faits.

Décision

Cette section décrit notre réponse à ces forces et contraintes. Elle est formulée par des phrases complètes et une voie active : *Nous allons ...*

Etat

Une décision peut être *proposée* si les parties prenantes du projet ne se sont pas encore accordées dessus pour l'instant, ou *acceptée* une fois qu'un accord a été trouvé. Si un ADR postérieur change ou revient en arrière sur la décision, elle est marquée comme *dépréciée* ou *remplacée* avec une référence à l'ADR qui la remplace.

Conséquences

Cette section décrit le contexte qui résulte de l'application de la décision. Toutes les conséquences et impacts devraient être listés ici, pas seulement ceux percus comme *positifs*. Une décision peut avoir des conséquences positive, négative ou neutre, mais toutes ces conséquences affecteront l'équipe et le projet dans le futur.

L'ensemble du document ne devrait pas être plus long qu'une ou deux pages. Les ADR seront écrits comme s'ils étaient une conversation avec un futur développeur. Cela requiert un bon style d'écriture, avec des phrases complètes et bien formées organisées en paragraphes. Les listes dans un style télégraphiques sont acceptables uniquement pour des questions de rendu visuel, pas comme une excuse pour n'écrire que des fragments de phrase. (*Bullets kill people, even PowerPoint bullets*.)

Etat

Acceptée

Conséquences

Un ADR décrit une décision structurante ou significative pour un projet spécifique. Cela devrait être quelque chose qui a un effet sur la manière dont la suite du projet va se dérouler.

Les conséquences d'un ADR vont probablement constituer le contexte d'ADR ultérieurs. C'est également similaire à l'idée du langage de *pattern* de Christophe Alexander : les réponses à grande échelle crée des espaces pour que celles à plus petite échelle s'insère dedans.

Developpeurs et les parties prenantes du projet peuvent voir les ADR, même si la composition de l'équipe change au cours du temps.

La raison derrière les décisions précédentes est visible pour tout le monde, présentement et dans le futur. Personne n'est laissé à se gratter la tête pour essayer de comprendre *Mais à quoi pensaient-ils ?* et le moment de changer des décisions anciennes pourra être clairement déduit des changements de le contexte du projet.

Retour d'Expérience

Vous aurez peut-être remarqué que ce billet est formaté comme un ADR lui-même. Nous avons utilisé ce format sur quelques uns de nos projets depuis Août 2011. Cela ne représente pas une plage de temps très longue (**NDT** : le billet a été publié en novembre 2011) mais les premiers retours tant des clients que des développeurs ont été clairement positifs. Durant cette période, nous avons eu de 6 à 10 développeurs qui ont tournés entre les projets utilisant des ADR. Tous ont témoignés qu'ils ont apprécié le niveau de contexte que leur donnait leur lecture.

Les ADR ont été particulièrement utiles pour capturer les intentions de conception sur le long terme. Nous avons plusieurs clients qui sont en train de stabiliser leur système actuel mais envisage une ré-architecture plus importante dans un futur relativement proche. En couchant ces intentions dans des ADR, nous ne rendons pas par inadvertance ces futurs changements plus difficiles.

Une objection potentielle est qu'en maintenant ces documents en gestion de configuration on les rend moins facilement accessible pour les chefs de projet, pour les clients et pour tous les autres qui n'utilisent pas quotidiennement la gestion de configuration comme l'équipe de développement le fait. En pratique, nos projets vivent quasiment tous dans des dépôt *GitHub* privés, de sorte que nous pouvons échangés des liens vers la dernière version dans le *master*. Puisque *GitHub* traite automatiquement le *Markdown*, c'est aussi convivial que n'importe qu'elle page de wiki.

Jusqu'à présent, les ADR ont démontré qu'ils étaient un outil utile, nous continuons donc à les utiliser.

Compléments d'informations

Lectures supplémentaires

Merci à Philippe Krutchen pour avoir discuter de l'importance des décisions d'architectures. Il m'a été signalé qu'il y avait d'autres discussions de ce type dans **Documenting Software Architectures** qui est dans le haut de ma liste de lecture.

Note sur la traduction

Ce document est une traduction en français que j'ai réalisé du billet de blog **Documenting Architecture Desisions** de Michael Nygard en date du 15 novembre 2011. Le billet original se trouve sur le blog de **Think Relevance** à l'URL.