1 Introduction

1.1 Titre

Je vais présenter ma thèse intitulé "Développement évolutionnaire de SdS avec une approche par patron de reconfiguration"

1.2 Diapo 2 : Les SdSs

Les services de secours. Les composants sont les services de secours. Chacune d'elle possède sa propre indépendance managériale puisqu elle dispose de ces propres ressources et définit elle même sont organisation interne. Chaque système possède sa propre indépendance oéprationnelle car ils peuvent réaliser leur mission indépendamment des autres. Les pompiers peuvent faire de lutte contre les incendies et faire du secours à personne sans l'aide des autres. Par contre l'intéraction avec les autres systèmes peut améliorer la mission de chaque service de secours. Par exemple, la police peut sécuriser et faciliter l'accès des pompiers à un sinistre.

Ce qui nous a particulierement interessé est la conséquence de l'ensemble de ces catéristiques qui est le développement évolutionnaire du sds.

1.3 Diapo : Le dev. évolutionnaire

On a considéré que le développement évolutionnaire est l'évolution permanente des buts et fonctions du SdS pendant sont execution.

La cause du développement évolutionnaire est qu'au moment de la conception, le SdS ne connaît pas toutes les informations sur l'environnement d'execution au moment de la conception.

E.g le sdis 56 n'a pas assez de ressources. E.g le les informations ne sont pas exacte, le sinistre à mal été évalué depuis le début.

Les raisons peuvent être est managériale, les consituants évoluent en changeant eux mêmes est les services fournit dans SdS sont impactés. Le SdS doit s'accomoder des capacités de ses constituants qui peuvent changer.

Des SdSs sont formés momentanement en réaction à un événement dont toutes les informations ne sont pas connu au moment de la conception et découvertes seulement pendant la phase d'execution.

Pour toutes ces raisons, le SdS est aménéà évoluer régulierement après la phase de conception et de déploiement. Ce qui nécessite des outils particuliers pour assister l'architecte du SdS afin de maîtriser au mieux le développement évolutionnaire.

1.4 Diapo : reconfiguration dynamique

La reconfiguration dynamique consiste donc à apporter des modifications sur un système pendant qu'il s'execute. Ces modifications peuvent être corrective, fonctionnelle si il s'agit d'ajouter une fonctionnalité, non fonctionnelle si elle modifie la qualité du système. La difficulté réside dans la maîtrise des parties du système interrompue, les dégradations de services, l'intégrité du système pendant le temps de la reconfiguration.

De manière générale, la reconfiguration dynamique vient du besoin de pouvoir appliquer des modifications à un système sans le redémarrer complétement. Dans un contexte SdS, c'est particulierement intéressant car les SdSs sont typiquement des systèmes qui une fois déployés ne peuvent pas être arretés puis redemarrés. Les risques seraient suivant le sds : économique et/ou humaine.

On a pris comme artefact de base pour raisonner sur les reconfigurations d'un système, ça description en terme de composant et connecteur. L'intérêt pour la reconfiguration et de pouvoir raisonner sur les parties modulables du SdS. cela permet d'identifier les parties sujettes au changement et de voir leurs dependances dans le SdS. On remarque que les opérations qui composent les scripts de reconfiguration peuvent être décomposées en opération primitive et organisées suivant l'importance des modications réalisées sur le SdS. Elles sont introspectives quand elle l'opération sert à oberser l'état du SdS ou intercessives quant elles réalisent des modifications sur le système. Les opérations sont : - ... - ... La conception d'une reconfiguration consiste à produire un script composé de ces opérations primitives qui permettent au système d'integrer les modifications souhaitées.

A travers la mise en œuvre de ces opérations l'architecte cristalisent un certain nombre de décision de conception qui ont un impacte sur la qualité des services pendant la reconfiguration et la préservation de la cohérence des transactions. Si on regarde un cas typique de stratégie de reconfiguration qui est la quiescence. L'architecte veut mettre à jours un composant sans interrompre les parties du système qui ne sont pas dependantes de ce composant. L'architecte va décider de déconnecter progressivement les composants qui dépendent du composant à remplacer. Puis une fois que ts les composants dépendants sont déconnectés, le composant ciblé par la reconfiguration peut être remplacé sans rédémarrer complétement le système n'y perdre les transactions démarrées.

Ce qui peut caractériser un problème de reconfiguration pour les sdss est :
- le contexte de reconfiguration (les opérations de reconfiguration disponibles) est amené à évoluer.
- mais aussi l'hétérogénéité des mécanismes de reconfiguration disponibles au moment de la reconfiguration.

Ce qui diverge des systèmes distribuées puisque souvent les composants implémentent les mêmes mécanismes de reconfiguration et n'évolue pas. Cela introduit de la variabilité également dans les décisions que peut prendre l'architecte pour réaliser une même stratégie de reconfiguration. Si on reprend l'exemple de la quiescence, l'architecte peut décider une approche où le composant ciblé par la reconfiguration atteind son état de quiescence naturellement ou alors il peut décider suivant les mécanismes disponibles de forcer l'état de quiscence pour terminer la reconfiguration.

Comme solution à la reconfiguration dynamique nous proposons d'outiller avec des patrons de reconfiguration et un processus de reconfiguration pour l'assister face au développement évolutionnaire des SdSs.

1.5 Diapo 6 : Solution - patron de reconfiguration dynamique et processus de conception

Ce qui nous intéresse est de documenter ce type de décision de reconfiguration pour la rendre réutilisable. Il est intéressant de rendre réutilisable. Pour adresser un problème de reconfiguration l'architecte doit être capable de communiquer ses choix de conception dans un but de validation par ces pairs pour améliorer et valider une reconfiguration. Ensuite les solutions doivent être capitalisées pour résoudre plus facilement des problèmes similaires déjà résolus. C'est un gain de temps et de fiabilité des solutions de conception proposée. mêmes erreurs.

Comme solution on propose la notion de patron de reconfiguration dynamique, que l'on a définit comme la documentation de solution à des prblèmes récurrents de reconfiguration dynamique.

Cette notion est un premier pas pour prendre en compte la variabilité des décisions de reconfiguration.

Les patrons de reconfiguration ne sont pas utilisables tels quels. En effet, on pense qu'en pratique l'architecte sera amené à définir des phases préliminaires à l'application de reconfiurraiton. En somme l'application d'un script de reconfiguration impliquera d'autres reconfigurations. nous avons documenté les étapes de conception qui permette l'élaboration d'une reconfiguration à partir de plusieurs patrons de reconfiguration. On a donc proposé un processus de conception pour la conception de reconfiguration dynamique.

2 Patron et processus de reconfiguration

2.1 Diapo : Comparaison de critère de réutilisation des décisions de reconfiguration

Pour définir des critères de documentation réutilisables, on a fait la synhtèse des parties descriptives des patrons de conception dans l'état de l'art. On en en tirer la liste suivante.

On peut classifier le type de documentation suivant son niveau de formalité.

Si on regarde la proposition de Allen, il contribue à la réutilisation en considérant la phase de reconfiguration comme un aspect à part dans le processus de developpement d'un système et donc en proposant un langage et des outils dédiés à la reconfiguration dynamique.

— Ce qu'on a identifié comme le contexte de la reconfiguraiton est une description formelle du style architectural.
— Ce qu'on a identifié comme le problème sont les invariants à respecter pendant la reconfiguration.
— La solution est une description des événéments qui déclenchent des ensemble d'opération de reconfiguration.

Ce qui limite la réutilisation de la documentation c'est l'absence d'une séparation claire entre ce qui est de l'ordre du contexte et de la problématique, ensuite le niveau de formalisme utilisé est très formelle ce qui limite la comprension globale de la documentation.

Si on regarde en détaille la proposition de Oliveira. Si on regarde les aspects réutilisations, elle concerne la description formelle de primitive de reconfiguration récurrente qu'elle a identifié comme des patrons de reconfiguration.
— Le titre des patrons correspond au résultat de l'application du patron comme la suppression d'un ensemble de canaux de communication.
— Le contexte est décrit par la description formelle des primitives de reconfiguration utilisé dans la solution
— Le problème correspond à la description de l'archicture initaiale et ciblé
— Ensuite la solution décrit les primitives de reconfiguration appelée

Ce qui limite la réutilisation est une granularité trop fine qui caractérise que des cas triviaux de reconfiguration.

D'autre contribution comme Gomaa base la documentation sur des descriptions semi-formelles. La contribution de réutilisation est la documentation de solution de quiescence dans une différent contexte architecturaux.
— Le titre du patron capture la contexte d'utilisation du patron., ici c'est le style architectural dans lequel est mis en œuvre le patron.
— Le contexte, documente le style architectural en décrivant le type des composants et la nature de leurs interactions avec les autres composants avec un diagramme de collaboration.
— La solution décrit avec un diagramme d'état modélisant l'effet de la reconfiguration sur les composants du système.

Ce qui limite la réutilisation
— Le titre ne permet de choisir un patron en fonction du type de solution.
— Ensuite le choix d'un diagramme de collaboration n'est pas adapté au contexte sds, puisque la taille des SdSs rend inutilisable ce type documentation.
— La description de la conséquence du patron n'est pas décrit séparement de la solution.
Par contre on a trouvé pertinent l'utilisation des diagrammes d'état pour modéliser les événéments de reconfigraiton, les opérations et les états des compsoants.

2.2 Diapo : Réutilisation des décisions de reconfiguration

Dans un contexte, SdS l'architecte doit avoir une documentation qui satisfait ts les critères de réutilisation ce qui permet de prendre en compte l'hétérogénéité et l'évolution de l'environnement qui est le résultat de l'indépendance opérationnelle et managériale des systèmes constituants.

Pour cela, on définit explicitement le contenu d'une documentation de reconfiguration.

De façon classique, un titre et une intention qui permette à l'architecte de donner l'intuition du principe mis en œuvre par la solution documentée. En général, le titre correpond à une métaphore du principe suivi par la solution et l'intention explique pourquoi une solution naive ne fonctionne pas et quel est un principe plus adapté.

pour documenter le contexte de reconfiguration, on s'est intéressé à la documentation du style architectural par un diagramme de block. Du point de vue de la reconfiguration, c'est une abstraction qui modélise les dépendances en tre les composants et les contraintes d'intérations.

Par rapport à Allen et Oliveira, on a exclu l'utilisation de langage formel qui n'aide pas à la compréhension. On a suivi un approche semie formelle avec l'utilisation de langage graphique. Par rapport à Gomaa, on a exclu l'utilisation de diagramme de collaboration qui ne sont pas utilisables du fait de la taille des SdSs.

Pour la description du problème, on a décrit quelle est le résultat de l'application de la reconfiguration. Dans notre documentation, on a documenté le problème par la description de l'architecture ciblée par la reconfiguration. Cette architecture est biensure liée à au contexte décrit avnt. On a documenté la reconfiguration initiale et ciblée par la reconfiguration par un digramme de block qui modélise le changement. Par rapport à allen, cet aspect est absent dans la documentation, oliveira décrit cet aspect mais avec un niveau de formalisme trop bas puisqu'elle documente le composant des connexions indépendamment d'un contexte plus globale qui est celui du des systèmes constituants qui sont impliqués par la reconfiguration. Par rapport, à gomaa, c'est une aspect nouveau puisque le problème est implicite à la solution. Ensuite, on a documenté quelles étaients les invariants liées à la reconfiguration. de façon informelle. Par rapport à l'ensemble des documentations existantes, on a documenté explicitement les forces (les principes) de la solution pour mieux la comprendre.

Pour finir on a documenté les forces qui doivent être mis en œvre dans la solution. Ce qui permet à l'architecte de comprendre ensuite qu'elles sont les principes de la solution.

Pour documenter la solution, on a exclu des réprensations formelles. On s'est également concentré sur l'utilisation de langage semi-formel. On a trouvé que l'utilisation de diagramme d'état comme utilisé par gomaa donne une niveau de détaille suffisant pour documenter une solution de reconfiguration. On l'accompagne d'une description textuelle qui définit les états, opérations et événements utilisées par la patron.

Pour finir, on a documenté en plus des autres type de documentation, les conséquences. Il s'est agit décrire, les conséquences du patron sur les aspects non fonctionnels du système. mais aussi l'influence des choix d'implémentation sur la reconfiguration. et suggérer des implémentations particulière.

Ensuite pour continuer notre approche de reconfiguration, on veut s'intéresser au processus de reconfiguration que l'architecte a à sa disposition dans l'état de l'art.

2.3 Diapo : Patron de reconfiguration - amélioration de la ré-utilisation

Dans la continuité du titre l'intention doit aboutir à la comprenhension de la métaphore. en expliquant, un approche naive et ses inconvénients puis on explique l'idée générale du patron et ses avantages en comparaison avec l'approche naive. l'approche du patron

On a constaté qu'une solution consistait à appliquer des reconfigurations graduellement.

On a expliqué d'abord l'intention de la reconfiguration sur la base d'un exemple. Ici, on a pris un système de service de secours. Dons le managanement d'une opération est géré par une codis qui doit déléguer le management à un autres CS.

On y explique la solution naive qui est d'attendre la fin des opérations en cours par les systèmes opérationnels. Dans ce cas, la reconfiguration a aucune change d'aboutir rapidement puisque le codis et devrait avoir pour résultat une forte dégradation de service.

La solution proposé dans le patron est de déployer directement le nouveau composant. Les systèmes opérationnels terminent leurs transaction en cours avec le codis puis démarre les nouvelles avec le poste de commandement mobile. Dans ce cas, les deux systèmes coexistent de façon concurrente et évolution donc conjointement.

2.4 Diapo : Patron de reconfiguration - délimitation du problème

Le problème de reconfiguration se réduit à la description du passage d'une architecture à une autre. On peut cependant, raffiner en différent problème suivant les solutions dévéloppés et en abstraire les régularités pour définir des patrons de reconfiguration.

On va essayer de déduire l'ensemble des besoins que cherche à satisfaire la solution pour capturer le problème exprimer par le patron. On a trouvé qu'un bon moyen est d'adord d'expimer l'architecture intiale, puis celle ciblée. L'architecture est bon support pour la compréhension du problème lié à qualité de service puisqu'elle modélise les dépendances fonctionnelles enntre les composants.

Ensuite on propose de documenter les contraintes que doivent satisfaire la solution. Elles peuvent être structurelles ou comportementales.

Les forces doivent expliquer les principes mis œvre dans la solution. Cela doit aider l'archticte à comprendre quelles sont les points importants de la solution.

Pour documenter des patrons de reconfiguration on a analysé des solutions récurrentes. Pour en abstraire les régularités qui vont former une patron de reconfiguration à documenter.

2.5 Diapo : Patron de reconfiguration - rédaction de la solution

Ensuite la solution consiste à décrire les mécanismes d'évolution et d'une grammaire de reconfiguration. La grammaire de reconfiguration consiste à décrire les opérations de reconfiguration et leur ordre afin de d'execution. Pour documenter la grammaire de reconfiguration nous nous sommes appuyé sur des diagrammes de collaboration et d'état ainsi qu'une description textuelle.

On a documenté la reconfiguration à partir du schéma ... et d'une description textuelle ... qui indique la sémantique des opérations.

2.6 Patrons de reconfiguration supplémentaires

A ce stade on a identifié et documenté plusieurs patrons de reconfiguration. On a donc la

- quiescence : dont l'objectif le principe est de rendre passif, les composants dépendant du composant ciblé par la reconfiguration.
- tranquilité : est une variante de la quiescence qui assouplie les critères de reconfiguration. Elle décrit les conditions dans lesquelles la reconfiguration pêtre réalisée sans attendre l'état de tranquilité
- co-evolution : contrairement à la quiescence et tranquilité, à déployer directement la nouvelle version du composant ciblé par la reconfiguration. La conséquence est que les deux versions d'un composant s'execute simultanément.
- opportuniste : par rapport aux autres, il concerne une granularité différente puisqu'il concerne seulement une opération de déconnexion et connexion qui est réalisé dès que l'occasion se présente.

3 Processus de reconfiguration

3.1 Processus de reconfiguration existant

Une fois que l'on a documenté des solutions de reconfiguration réutilables on s'est interessé au processus de conception.

Une approche dans la conception des reconfigurations est automatique. L'architecte fournit l'architecture ciblée et la reconfiguration est dérivée automatiquement. C'est le cas de la proposition de Boyer. On voit que l'algorithme de reconfiguration se compose de deux phases, la prémière phase réalise ttes les opérations de déconnexion, d'arêt et de suppression de composant, puis la seconde phase réalise toutes les opérations de connexion, création et démarrage. Ensuite si on détaille les opérations d'arêt, son arêt impliquent un comportement récursif qui propagent l'opération d'arêt à tous les composants qui dépendent du composant ciblé par l'opération d'arêt.

Dans d'autres cas, l'approche est manuelle, dans la contribution de buisson, l'architecte doit fournir une reconfiguration et une preuve vérifier pour apppliquer la reconfiguration. Comme expliqué par l'auteur, le processus de reconfiguration est itératif, car en pratique l'architecte fait plusieurs itérations, soit pour refaire la reconfiguration ou soit pour refaire la preuve de sa reconfiguration.

De façon général, on voit dans l'état de l'art que les processus de reconfiguration sont plus ou moins artisanals. Comme dans un contexte SdS, l'architecte est obligé d'adapter ses hypothèses de reconfiguration à cause de l'indépendance des CSs. Dans ce sens, pour aider l'architecte à réaliser des reconfigurations on a proposé un processus où l'on a identifier les étapes de la conception d'une reconfiguration.

3.2 Proposition d'un processus de reconfiguration

A partir de l'étude des pratiques de reconfiguration, on a proposé un processus de reconfiguration décomposé en trois phases :

- préparation : opérations de reconfiguration préalable au opération. Il s'agit d'ajouter des mécanismes d'évolution comme l'ajout de tampons par exemple ou mécanisme de synchronisation.
- modification : action souhaité intialement. les oéprations de reconfiguration qui vont permettre les modifications souhaités par l'architecte.
- nettoyage : suppression des mécanismes d'évolution liés à la reconfiguration qui ont été déployés dans la phase de prépration.

Ensuite on a considéré que chaque phase de prepration peut être également décomposé en phase : preparation, modification et nettoyage.

3.3 Utilisation des patrons de reconfigiration

Ici l'intérêt d'expliciter la récursion dans le processus est de permettent à l'architecte de contrôler la qualité de service explicitement pendant la reconfiguration en décidant de relâcher des contraintes pour permettre la reconfiguration. C'est dans le cas, où les contraintes de reconfiguration rendent impossible la reconfiguration. Dans ce cas l'architecte peut décider d'une nouvelle itération dont l'architecture ciblé et l'architecture intégrant les contraintes suffisantes pour appliquer les reconfiguration décidé dans l'itération précédente.

Le processus s'appuie sur les patrons de reconfiguration. Dans un premier temps, l'architecte analyse les modifications à apportées.

il utilise le catalogue de patron pour identifier les problèmes qu'il peut rencontrer et comment les résoudre. ou il définit un nouveau patron.

Dans le cas, ou le solution n'est pas directement applicable alors il peut décider d'une nouvelle itération. Dons la cible est l'archicture requise pour appliquer la reconfiguration de l'itération précédente.

4 Expérimentation

4.1 Description de l'expérimentation

Pour vérifier la fesabilité de notre approche on a proposé une expérimentation.

Pour vérifier la fesabilité de notre approche on a proposé une Pour cela, on a développé un framework de reconfiguration dans le but de simuler l'execution d'une sds reconfiguré. Le framework embarque des opérations de reconfiguration et mécanismes de reconfiguration à disposition de l'architecte pour décrire le script. Ensuite le framework peut être configuré avec l'architecture à vérifier pendant la reconfiguration et la description de scénario d'execution.

On a fourni la description d'une architecture de SdS et d'une évolution à déployer. Pour cela on a modélisé deux configurations de l'étude de cas que nous avons étudié dans cette thèse. L'objectif de la reconfiguration était de déléguer le commandement d'une ensemble de système constituant. Les propriétés vérifiées étaient que l'ensemble des opérateurs restent sous la supervision d'un organe de contrôle et qu'à la fin de la reconfiguration l'état de l'avancement de la mission ne soit pas perdu.

Pour vérifier la fesabilité de notre approche on a proposé une Pour aider l'architecte de la reconfiguration nous avons fourni l'ensemble des patrons de reconfiguration dynamique ainsi que la description de processus de reconfiguration dévéloppé dans la thèse.

4.2 Interprétation et limite

Pour réalisé la reconfiguration, l'architecte a réalisé trois itérations en utilisant le patron de co-évolution et de tranquilité. L'execution de la reconfiguration a respecté les objectifs de la reconfiguration.

On a essayé d'évaluer la réutilisation des patrons de reconfiguration en vérifiant que l'architecte les réutilisent dans la conception de son script. On s'est posé la question de savoir si l'architecte à pu envisagé plusieurs solution de reconfiguration, choisir la plus adapté à l'objectif de reconfiguration et à réussie à l'implémenté.

On pense à ce stade que notre proposition assiste l'architecte dans le developpement évolutionnaire du SdS. Cependant des points de l'expérimentation sont à améliorer. Notamment le choix des architectes, dans cette expérimentation, nous n'avons pas eu d'autres sujet que nous même. Et ce stade, les scénarios d'execution du SdS sont limités et modélise des cas d'execution simple.

5 Conclusion et Synthèse

5.1 Q1.

5.2 Conclusion

On a proposé d'assister le developpppement évolutionnaire des SdSs par de la reconfiguraiton dynamique.

Pour cela on a d'abord montré que -> les documentations de l'état de l'art était limité, -> on montrer le caractère artisanal dans la conception des reconfigurations

On a proposé une expérimentation -> developpé un framework de reconfiguration simulant l'exection d'un SdS et de reconfiguration dynamique. -> On a modélisé un cas d'étude comportant des reconfigurations à réaliser -> Cela nous a permi de montrer le caractère réutilisable de notre documentation -> mais aussi l'utilité de processus de reconfiguration pour assister l'architecti en le guidant et l'aide à maîtriser la dégradation de service pendant la reconfiguration.

5.3 Perspectives

a court terme on propose d'améliorer l'expérimentation. L'expérimentation est limité dans la mesure ou les sujets ne sont pas externent au travaille de conception des patrons de reconfiguration et du processus. Ensuite l'étude de cas n'est pas complète.

On proposerait d'améliorer la modélisation de l'étude de cas en impliquant les systèmes consituants réèlle.

Ensuite, on souhaiterais pouvoir réaliser deux populations ...

Ensuite on souhaiterai intégrer dans notre environnement de reconfiguration un outil pour permet le suivi des décisions de reconfiguration. On envisage par exemple, la possibilité d'annoter un ensemble d'opération de reconfiguration correspond à une décision de reconfiguration.

On pourrait envisager ensuite l'étude de la composabatilté des patrons de reconfiguration. Sur la base de la fréquence d'utilisation des patrons de reconfigurations conjointe pour pourrait en déduire des catalogue de patron de reconfiguration pour des domaitnes en particluer.

A ce stade, la sûreté, c'est à dire la garantie qu'ils vont réaliser ce qu'on attend, tiens sur le fait que ses patrons sont produits à partir des expériences tirées de leur mise en œvre.

Les méthodes de vérification sont couteuse en temps de conception ou de calcule. Le caractère réutilisable des patrons sont p-être une solution pour compenser le coût. Des méthodes de vérification que l'on envisage sont par exemple la preuve de théorème. peut impliquer des preuvent des grandes donc couteuse en ressource humaine. les méthodes de modèle checking pose habituellement des problèmes de temps de calcule.

Pour finir, dans cette thèse on pas pris en compte la spécificité des systèmes intégrant des être humains qui est une caractéristique des SdS. La queston qu'on ce pose est comment intégrer dans notre documentation les contraintes liées aux aspects sociaux propre. devrait intégrer une description sociale impliqué dans la conduite des changement