# 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

Des exemples de SdS sont par exemple : Des systèmes de surveillance d'inondation ou les capteurs collaborent momentannement pour calculer des informations environnementales

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 oépérationnelle 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.

Internet où les composants sont des routeurs qui implémentent le protocole ip et collabore volontairement

Le Web ou les composants sont les services web dont la composition fournissent d'autres services.

Les SdSs sont une classe système dont les constituants sont eux mêmes des systèmes. Les SdSs héritent des caractéristiques des systèmes distribués car leur composant sont géographiquement distribué. Mais ce qui les différencie de système distribué est que leurs consistuants possèdent leur propre indépendance managérialle et opérationnelle. La raison de la collaboration entre ces systèmes est le comportement emergent produit. C'est un comportement qui ne peut pas être réalisé par un seul des constituants du SdSs et qui apporte en général un bénéfice à tous les participants au SdS.

Parmis les SdSs, on peut distinguer plusieurs catégories de SdS suivant le niveau coercitif sur ses composants à prendre en compte dans leur ingénirie. Les réseaux de capteurs malgrès qu'ils peuvent opérer indendamment du SdS ont des objectifs de collaboration imposés.

Les services de secours sont plutôt un SdS consensuel car les objectifs du SdS sont décidés de façon consensuel.

Internet définit des objectifs de routage mais l'implication des composants reste volontaire.

Le web les objectifs ne sont pas idenifier et la collaboration est la collaboration est volontaire.

Dans la thèse, on a étudieé plutôt les systèmes consensuels. Sans donner un importance particulière à la catégorie de SdS. 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 3

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.

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 4

Comme solution les SdSs sont amanés à intégrer régulierement de plus en plus de logiciel pour supporter le developpement évolutionnaire et en particulier au niveau de la phase de conception de l'architecture des SdSs.

Dans notre thèse on a pris comme référénce deux projets européens qui sont DANSE et COMPASS. On peut remarquer la pré-pondérance du logiciel dans le processus de conception de l'architecture des SdS. Les phases de conception intègre des générateur d'architecture, des outils de simulation pour la vérification et divers langage pour modéliser les objectifs du SdS, les contraintes à respecter dans les architecture et la description du comportement des systèmes consitituants.

Un aspect du cycle de vie des systèmes moins étudié pour les SdSs est la phase de déploiement de la nouvelle architecture dynamiquement et que nous avons abordé en particulier dans cette thèse à travers le problème de la reconfiguration dynamique.

## 1.5 Diapo 5

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.

Comme solution à la reconfiguration dynamique nous proposons d'outiller l'architecte pour l'assister face au développement évolutionnaire des SdSs.

## 1.6 Diapo 6

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 d'intégrer la notion de patron de conception bien connu dans la domaine logiciel. Les patrons de conception sont une solution documentée face à un problème récurrent. Les patrons de conceptions les plus connus sont ceux du GoF, ils sont utilisés dans la programmation orientée objet pour favoriser la réutilisation du code. L'exemple est repris dans d'autres champs du logiciel comme la conception d'architecture pour intégrer des aspects non fonctionnelles comme la scalabilité ou l'adaptabilité.

Une première difficulté a été l'étude de SdS pour identifier des problème de reconfiguration que l'on pourrait rencontrer lors de reconfiguration. Dans l'état de l'art des SdSs peu d'étude de cas se sont intéressées à capturer des configurations résultantes d'évolutions successives et des propriétés de reconfiguration souhaitées.

## 1.7 Diapo 7

Le challenge à été de capturer des configurations avec le niveau de précision suffisant pour capturer des propriétés de reconfiguration à étudier mais aussi avec un niveau d'exactitude correcte pour capturer des contextes de reconfiguration.

Une dernière difficulté à été de determiner le processus de reconfiguration à utiliser pour mettre en œuvre le concept de patron de reconfiguration.

### 1.8 Diapo 8

Dans l'état de l'art on a constaté que la reconfiguration dynamique a été principalement étudié du point de vue des langages pour la décrire, des propriétés des protocoles de reconfiguration et l'automatisation des protocoles de reconfiguration.

## 2 Framework pour la modélisation des SdSs

### 2.1 Diapo 12

**à déplacer** On a vu qu'une problématique de cette thèse a été la modélisation d'architecture de SdSs nécessaire à l'étude de la reconfiguration dynamique. Le besoin ... Les modèles d'une manière générale sont des abstractions de la réalité. En génie logiciel, il capture l'architecture du SdS. Ils sont utilisés pour améliorer la qualité et le cout des systèmes. On peut réduire les cout et augmenter qualité par exemple avec l'utilsiation des modèles pour générer automatiquement l'architecture d'un système, faire des simulations pour améliorer la qualité de l'architecture avant de la déploiyer.

Un objectif de modélisation des SdSs est d'obtenir des modèles exactes et précis. Ce qui nous interesse est de réussir à modéliser les abstractions qui concour à la réalisation du SdS. On s'est interessé en particulier à identifier tous les CSs et les ressources qu'ils déploient ainsi que leurs objectifs d'interaction. On aura besoin de modèle suffisament précis pour modéliser les décisions architecturales du SdS.

La problématique est un contexte de modélisation difficile dû aux frontières du SdS flou. Les frontères sont flou car les composants du SdS sont développés par des tiers. Si on regarde la figure, si le SdS définit ses frontières aux composants A,B, et C, il ne prend pas en compte D qui si il se désengage de A et C peut dégrader fortement le fonctionnement global du SdS.

Parmis les études de cas disponibles peu s'intèressent à la reconfiguration dynamique mais plutôt à la conception des architectures. On s'est plutôt intéressé au langage et processus de modélisation mis en œvre dans les études de cas. Pour la conception des systèmes complexes comme les sdss les langages de modélisation ont été développé pour gérer la compléxité des modèles, la précision et l'exactitude. Dans cette section, nous allons étudier les langages de modélisation dans le developpement des SdSs et leur portée. On définira leurs caractéristiques utiles et comment elles sont exploiter dans les conceptions des SdS à travers deux projets Europééens qui sont DANSE et COMPASS.

### 2.2 Diapo 13

Un premier langage de modélisation auquel on s'est intéressé est SySML. C'est un langage de modélisation qui a l'ambition de permettre une approche de modélisation holistique. L'idée est que le langage puisse supporter toute les phases de développpement du SdS, notamment en intégrant tous les corps de métier qui peuvent intervenir dans le développement d'un système.

Les principaux éléments de modélisation utilisés sont la notion de block et de port. Les blocks modélisent des unités modulaires d'un système. Les ports décrivent les points d'intercation entre les blocs. Il y plusieurs catégorie de port qui peuvent permettre de modéliser des interactions entre des parties logicielles et matérielles. Ensuite les flux ajouter une description physique des interactions entre les blocs. Pour finir la relation d'allocation permet de modéliser des relations de raffinement entre les éléments modéliser ce qui est utile pour supporter les processus de modélisation descendant ou ascendant.

**à déplacer** Plusieurs type de diagramme interviennent : les diagrammes de block, d'interaction, d'activité, paramétrique, d'exgience.

Les avantages pour la modélisation des systèmes de systèmes sont la description des connectivités puisque la modélisation des SdSs s'interessent à la description des interactions entre les CSs, et traçabilité entre les éléments de modélisation qui permet de définr des relations entre des modèles developpé à différent niveau d'abstraction et/ou différent aspect, ce qui est un avantage des SdSs qui intègre différent parti prennant dans la modélisation des SdSs.

Evidemment l'inconvénient est l'absence de parti pris sur le type des éléments à modéliser. Des frameworks de modélisation Comme UPDM peuvent vu comme des profils de SySML est proposer des modèles pour le developpement des SdSs.

## 2.3 Diapo 14

un second langage que l'on a étudié est UPDM. UPDM est profil de SysML. Sa principale contribution est la proposition de 40 vue pour la modiélisation de SdS.

Différent type de vue propose de modéliser des niveau d'abstraction différent : - ..

Concretement on a regardé leur utilisation pour définir leur portée et comment ils sont utilisés. Pour cela on a étudié les principaux projets de recherche européen dans le domaine de SdSs. Il s'agit de DANSE. qui a basé sont projet developpemment de SdS sur le langage UPDM.

Le tableau fait la synthèse des vues et diagrammes selectionnées par danse.
Si on regarde leur utilisation, les vues opérationnelles servent à identifier ....
les vues systèmes servent à ...

après la phase de modélisation avec UPDM, les modèles sont raffinés vers des représentation plus formelles. définition des contrats sur l'architecture comme objectif à optimiser pour générer une architecture, la vue sv-1 assiste la génération d'architecture automatique. définition de contrainte de cout à respecter, ...
les vues sv-4 servent à analyse les evolutions de possible du SdS. définition de relation de causalité et probabilité de réalisation. + Définition de contrat sur les CSs

Si on regarde en comparaison le framework developpé par compass, les vues developpées sont similaire.

A ce stade on voit que l'exactitude est obtenu grâce à un processus de développement descendant. La modélisation démarre par des abstractions hauts niveaux qui capture les systèmes participants et les objectifs d'interaction. Puis raffine sont raffiné vers des abstractions concrètrs. Au niveau de la précision on a noté que SySML et UPDM ne sont pas suffisant pour exprimer des propriétés vérifibles de façon automatique.

## 2.4

Dans notre approche de modélisation des architectures de SdSs on a choisi de se baser sur l'existant. Sans proposer de modification des profondes. En effet, en comparaison avec d'autres travaux de modélisation que soit pas basé sur UPDM on voit que les vues et pts de vue sont similaires. On a pris en parti pri que cela suffit pour le moment à fournir des données réalistes.

On propose donc de suivre une approche top down pour la modélisation. - On a ajouté une phase de modélisation des exigences pour les structurer. - La phase de modélisation du niveau SdS repose sur les vues ov1, ov1b, ov5. - On a explicité une phase de validation manuel. - Puis une phase de modélisation de l'architecture au niveau CS. - Puis une phase de vérification de la cohérence de l'architecture avec une simulation.

Si on regarde la modélisation du SdS on a retenu les vues OV-1, OV-1b, OV4 et OV-5. On a rejoint le parti de DANSE, cependant on a trouvé plus pertinent l'utilisation du diagramme de cas d'utilisation pour modéliser les objectifs d'interaction entre les resssources. Cela améliorer la lisibilité du diagramme et permet de guider le developpement de la description des fonctions du SdS. Le détaille de la connectivité modélisé avec OV2 dans DANSE description de la connectivité est décrite dans la vue OV-4.

La phase de modélisation du niveau CS ... ajout de contrainte ocl pour préserver l'expressivité du langage on a opté pour l'utilisation de primitive architectural spécifié en ocl ce qui nous a permis de l'intégrer directement dans la vue sv-1, utilisation de la sémantique du pi-calcul.

A ce stade on a clarifié les choix de modélisation et les raisons. Maintenant je vais expliquer configurations que l'on a choisi de modéliser.

## 2.5 Diapo 17

On a choisi comme cas d'étude le système de service de secours d'urgence. Le domaine du système est un cas typique de SdS.

La mission principale sur laquelle se concentre notre cas d'étude est la protection des personnes, des biens et de l'environnement. Il peut s'agir du transport d'une personne à l'hôpital, de limiter la progression d'une inondation, d'un incendie ou de circonscrire une pollution fluviale. De manière générale, un des objectifs de la mission est d'empêcher l'évolution d'un sinistre et de revenir à une situation normale.

Les principaux cs sont :
- Les hôpitaux prennent médicalement en charge les victimes.
- Les pompiers, à l'échelle départementale, sont structurés par un Service Départemental d'Incendie et de Secours (SDIS) qui possède ses propres ressources matérielles et sa propre structure de commandement. Ce service est responsable du transport des blessés, de la lutte contre les incendies et de la protection des biens.
- Le Service d'Aide Médicale Urgente (SAMU) est le centre de régulation médicale d'urgence qui régule les ressources de soins d'urgence (ambulances par exemple).
- Enfin, la sécurité civile est impliquée dans des situations de crises majeures et peut fournir des moyens aériens.

Au niveau du territoire français, les services de secours sont organisés par département. Chaque type de service est déployé sur chaque département français avec son propre financement et sa propre structure hiérarchique de commandement. La collaboration entre les services est favorisée par le réseau Il s'agit du réseau numérique de communications intra-services (quand les communications sont limitées aux membres d'un même service) et inter-services (quand les communications permettent à des membres de services différents de communiquer). Les collaborations que peuvent former le sos ont des échelles différentes. Elles peuvent être :
- Locales quand il s'agit de communications entre deux ressources proches géographiquement.
- Départementales quand les ressources qui communiquent sont distribuées à l'échelle du département.
- Inter-départementales quand ce sont des ressources de services appartenant à deux départements différents.
- Nationales quand elles impliquent des membres du gouvernement.

Une gestion efficace consiste à prendre les bonnes décisions au bon moment. Cette efficacité dépend de la qualité des informations et du moment où elles sont fournies. Pour ces raisons les services de secours se structurent autours d'une chaîne de commandement. Les composants de la chaîne de commandement (qui sont les ressources des services de secours) sont partionnés suivant leur niveau de responsabilité décisionnaire. Ces niveaux de décision sont d'ordre :
- Stratégique. Cela consiste à définir les objectifs et anticiper les besoins opérationnels en réservant des ressources, afin de les allouer au moment opportun.
- Tactique. Cela consiste à assigner des objectifs aux ressources allouées pour réaliser les objectifs stratégiques.
- Opérationnel. Cela consiste à décider d'opérations primitives dans le but de réaliser les objectifs tactiques.

La discipline hiérarchique impose aux composants de n'interagir qu'avec le niveau n+1 ou n-1. Lorsqu'il s'agit d'interaction avec les composants du niveau n+1, ces interactions sont des remontées d'information. Par exemple, les composants du niveau opérationnel font une synthèse de la situation observée aux composants du niveau tactique. Les communications vers le niveau n-1 concernent des directives. Par exemple, les composants du niveau tactique décident de la zone de positionnement d'un composant opérationnel et des actions qu'il doit réaliser.

Cette chaîne de commandement se reflète dans la structure du réseau de communication des pompiers. La communication entre le CODIS et ses ressources déployées se fait principalement via un canal de communication radio appelé canal opérationnel par le réseau ANTARES. Le canal opérationnel est

un groupe de communication déployé pour chaque SDIS. Chaque membre du groupe connecté sur le même canal de communication peut écouter les messages envoyés par les autres membres du groupe. On parle d'écoute passive. Via le canal opérationnel, le CODIS gère toutes les interventions courantes en cours dans le département :

— Les communications se font seulement de façon verticale. Par exemple, un VTU ne peut pas communiquer avec un autre vtu.
— Les abonnés ne prennent la parole que s'ils ont l'autorisation du CODIS et s'il n'y a pas une autre communication en cours. Par exemple, le VTU demande l'autorisation de parler avant d'envoyer son rapport. Un seul abonné peut avoir la parole à un moment donné.

D'autres types de canaux sont disponibles comme le canal tactique et stratégique, pour les niveaux correspondants de la chaîne de commandement. Ces canaux fonctionnent de la même manière que les canaux opérationnels, avec les mêmes contraintes.

La première configuration considère que deux sdis ont une collaboration opérationnelle interdépartementale, c'est-à-dire qu'ils communiquent directement, seulement pour ce qui concerne les actions de terrain réalisées pour remplir la mission. Une telle collaboration est mise en place, par exemple, lorsqu'un SDIS fait face à un manque de personnel, et demande donc des ressources manquantes auprès d'un SDIS voisin pour une mission de petite taille. Dans notre scénario, cette configuration est utilisée lorsque les habitants appellent des services d'urgence en réponse à ce que l'on pense être une inondation localisée.

Dans la deuxième configuration, deux SDIS ont une collaboration tactique et stratégique interdepartementale, c'est-à-dire qu'une chaîne de commandement hiérarchique est mise en place. En plus de la collaboration opérationnelle, la collaboration tactique permet aux commandants de communiquer avec leurs subordonnés, afin de leur donner des instructions et de recueillir des rapports sur les actions sur le terrain. La collaboration stratégique garantit que le SDIS collaborateur adopte des orientations cohérentes afin d'atteindre les objectifs de niveau supérieur définis par la mission. Cette deuxième configuration est utilisée lorsque la crise est plus importante, ce qui nécessite plus de ressources que dans la première configuration, et lorsque ces ressources nécessitent une coordination étroite. Dans notre scénario, le SdS évolue vers cette seconde configuration lorsque les services d'urgence découvrent que l'inondation est plus critique qu'ils ne le pensaient initialement.

Enfin, dans la troisième configuration, un SDIS collabore avec le SAMU et la sécurité civile. Cette configuration se produit lorsque les services d'incendie et de secours ont besoin de l'aide d'autres services pour faire face à la crise. Dans notre scénario, le SAMU est impliqué afin de réguler l'évacuation des blessés vers les hôpitaux les plus proches, et la sécurité civile fournit des ressources de type hélicoptère lorsque l'inondation rend les routes inutilisables.

Maintenant que l'on a identifié des candidats à la modélisation, on peut appliquer notre framework de modélisation.

## 2.6 Diapo

étape 1

étape 2

étape 3

étape 4

étape 5

## 2.7 Diapo

A ce stade, nous avons modélisé 3 architectures qui correspondent à la description d'environ ... composant/connecteur. et xx diagrammes.

Le processus de modélisation est supporté par le langage UPDM. On a vu que UPDM n'est pas utilisable tel quel. il a fallu selectionné les vues les plus pertinantes parmi la quarantaine disponibles. Pour confirmer nos choix nous nous somme basé sur le choix de modélisation d'autre étude de cas.

Cela nous a permis d'expliciter des phases de modélisation utilise. La modélisation des architectures reposent sur l'utilisation de primitive architecturale ocl. L'avantage est d'ajouter la précision nécessaire à une vérification automatique des décisions architecturales. mais aussi d'assister la vérification de l'exactitude de l'architecture en rendant plus expressif les contraintes ocl.

Cependant on peut noter tout de même que les modèles ne sont pas complets et pourrait être plus détaillés.

# 3 Patron et processus de reconfiguration

## 3.1 Diapo :

Pour la reconfiguration, on s'est fixé comme objectif l'application de changement sur le sds pendant qu'il s'execute. D'après l'exemple, que l'on a developpé on va s'intéresser à maîtriser la qualité de service et s'assurer de la cohérence de certaine transaction pendant la reconfiguration.

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. 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 nous intéresse est de documenter ce type de stratégie de reconfiguration pour la rendre réutilisable. Sur la base du type de contenu des patrons de conception en général, un patron réutilisable est décrit par son intention, contexte, problème, force, solution et conséquence.

Ce qui peut caractériser un problème de reconfiguration pour les sdss est 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. 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.

Dans la suite, on va voir d'après l'état de l'art comment l'architecte peut documenter sa reconfiguration de façon réutilisable mais aussi quel processus d'ingénierie il a à sa disposition pour assister le travail de conception d'une reconfiguration quand les mécanismes de reconfiguration ne sont pas anticipables.

## 3.2 Diapo :

On peut classifier le type de documentation suivant son niveau de formalité. On a des documentations qui sont très formelles comme celles utilisées par Allen ou Oliveira pour documenter des reconfigurations. Ce niveau de formalisme permet de vérifier formellement qu'une propriété est vérifiée pendant la reconfiguration. Si on regarde les aspects réutilisations, on comprend l'intention de la reconfiguration en analysant l'architecture ciblée, le contexte est difficilement compréhensible si l'architecte ne connait pas le formalisme, la solution est documentée mais difficile à interpreter si on ne connait pas le formalisme, les forces et conséquences ne sont pas clairement documentées. La réutilisation dans ce cas est partielle.

Si on regarde des approches moins formelles. Les travaux de Gomaa sont plus intéressants. L'intention de la reconfiguration est implicite ici car les patrons ont tous pour objectifs de remplacer un composant avec une stratégie de quiscence. Le problème est également implicite, il s'agit préserver les

... (quiescence), par rapport au documentation précedente, on voit que le contexte est documenté par des digrammes de collaboration qui montrent le rôle des composants dans l'architecture. La solution est documentée via des diagrammes de collaboration qui renseignent sur le rôle des composants entre eux. Par rapport à l'approche précédente, la compréhension est simplifiée et elle ne nécessite pas de connaissance pointue.

### 3.3 Diapo :

A ce stade on voit que les approches comme celle de Gomaa sont du point de vue de la réutilisation, une type de documentation plus pertinent.

On par rapport au critère de reconfiguration SdS, on voit que voit que la solution de Gomaa est limité dans un contexte sds. en effet, on ne peut pas faire l'hypothèse que les composants implémentent strictement les mêmes mécanismes de reconfiguration. On veut proposer une documentation qui prend en compte plus de type de décision de reconfiguration. On voit que l'on aura besoin d'expliciter d'autres champs pour notre concept de patron de reconfiguration.

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.

### 3.4 Diapo : à déplacer

Dans un premier temps, cela consiste à considérer la reconfiguration comme un aspect à part dans le developpement des systèmes. Donc considérer des langages et outil propre au problème de reconfiguration. Ces langages assistent l'architecte pour maintenir des propriétés pendant la reconfiguration. On voit que ces méthodes n'aide pas vraiment l'architecte à trouver une solution mais plutôt à vérifier le script conçu.

Des approches qui automatisent la tâche de reconfiguration est de dériver automatiquement des scripts de reconfiguration à partir de la descrition de l'architecture cible et source ne permettent pas maintenir pendant une reconfiguration. Dans ce cas, les solutions de reconfiguration à traiter sont

## 4 Conclusion et Synthèse