Développement de grands systèmes avec Rational Unified Process

Maria Ericsson

Livre blanc de Rational Software

TP 156

Rational®
the software development company

Table des matières

Historique

Ce document est tiré de "Systems of Interconnected Systems", publié dans ROAD, en mai/juin 1995, par Ivar Jacobson, Karin Palmkvist et Susanne Dyrhage [1]. Il tire parti des entrées de plusieurs projets de développement de grands systèmes et doit s'aligner avec le processus RUP (Rational Unified Process) version 5.1 [2] et le langage UML (Unified Modeling Language) [3].

Ensembles de systèmes interconnectés

Le développement de grands systèmes représente une augmentation considérable de la complexité. Non seulement, vous devez être en mesure de comprendre un ensemble complexe d'artefacts, mais vous introduisez également une surcharge du fait que vous devez gérer un ensemble de ressources plus important. Ce document décrit un schéma d'architecture utilisé pour contrôler la surcharge de complexité ajoutée. Le schéma d'architecture, parmi d'autres espaces discutés à la section [4], est référé en tant qu'**ensemble de systèmes interconnectés**.

Cette construction s'avère utile lors de la création de systèmes très grands ou complexes, tels que des systèmes de commande ou de contrôle ou des solutions informatiques hautement intégrées. Ces types de "super systèmes" sont dans la plupart des cas divisés en plusieurs parties distinctes, chacune développée de manière indépendante comme un système séparé. Un super système est mis en oeuvre par un ensemble de systèmes interconnectés, communiquant entre eux pour remplir les tâches du super système. L'un de ces systèmes représente les capacités globales et peut être appelé **système principal**. Les autres systèmes représentent une partie de la totalité et peuvent être appelés **systèmes subordonnés**. Un système principal se distingue clairement des systèmes subordonnés qui le mettent en oeuvre. La relation entre les différents types de systèmes se distingue de la manière suivante : les systèmes subordonnés se qualifient de sous-systèmes, comme le montre la figure 1.

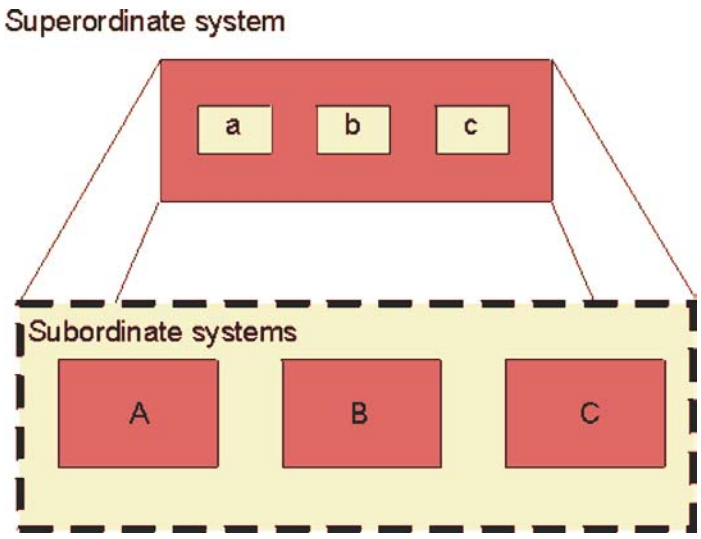

Figure 1. La spécification d'un système principal est mise en oeuvre par un ensemble de systèmes interconnectés, dans lequel les systèmes A, B et C sont respectivement des implémentations des sous-systèmes a, b et c du système principal.

La séparation du système principal de ses systèmes subordonnés présente plusieurs avantages :

- Les systèmes subordonnés peuvent être gérés séparément durant toutes les activités du cycle de vie, y compris les ventes et la livraison.
- L'utilisation d'un système subordonné est simplifiée pour mettre en oeuvre d'autres systèmes principaux en le reliant à d'autres ensembles de systèmes interconnectés.

- Lorsque vous commencez à construire un système, vous ne savez pas toujours s'il s'agit d'un ensemble de systèmes interconnectés ou non. Vous pouvez commencer à travailler avec une vue de système "simple" et décider relativement tard dans le cycle de vie d'appliquer ou non le pattern d'ensemble de systèmes interconnectés.
- Elle permet d'appliquer des modifications internes aux systèmes subordonnés sans développer de nouvelle version du système principal. De nouvelles versions des systèmes principaux sont requises uniquement dans le cas de modifications fonctionnelles majeures.

Chaque système subordonné dispose d'un ensemble d'artefacts qui lui est associé, offrant ainsi une traçabilité claire entre eux. Une fonction de trace est également gérée entre les ensembles d'artefacts des systèmes subordonnés et les ensembles d'artefacts correspondants du système principal. Chaque système subordonné peut être géré en tant que projet de développement distinct avec ses propres phases de cycle de vie : création, élaboration, construction et transition.

Si le "super système" que vous créez est très grand, il se peut qu'un système subordonné doive être subdivisé davantage encore et traité comme un ensemble de systèmes interconnectés.

Cycle de vie du développement logiciel

Dans le processus RUP (Rational Unified Process), le cycle de vie de développement est présenté et discuté selon deux perspectives : la perspective de gestion et la perspective de développement, comme le montre la figure 2.

Dans le cadre de la gestion, quatre phases de cycle de vie sont nécessaires au développement d'un système ou d'une nouvelle génération de système. Dans le cadre du développement, vous développez de manière itérative des versions du système qui sont de plus en plus complètes. Les activités à effectuer au cours d'une itération ont été regroupées dans le processus RUP (Rational Unified Process) sous forme de flux de travaux principaux. Chaque flux de travail principal se concentre sur la description d'un certain aspect du système, résultant en un modèle de système ou un ensemble de documents.

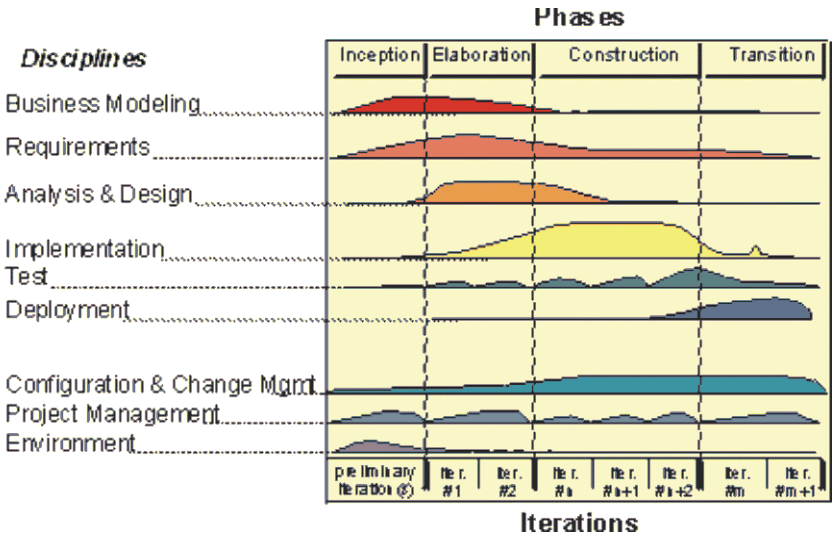

Figure 2. Modèle itératif

En appliquant cela aux ensembles de systèmes interconnectés, le système principal, ainsi que chacun de ses systèmes subordonnés, traversent leur propre cycle de vie et sont souvent traités comme des projets distincts.

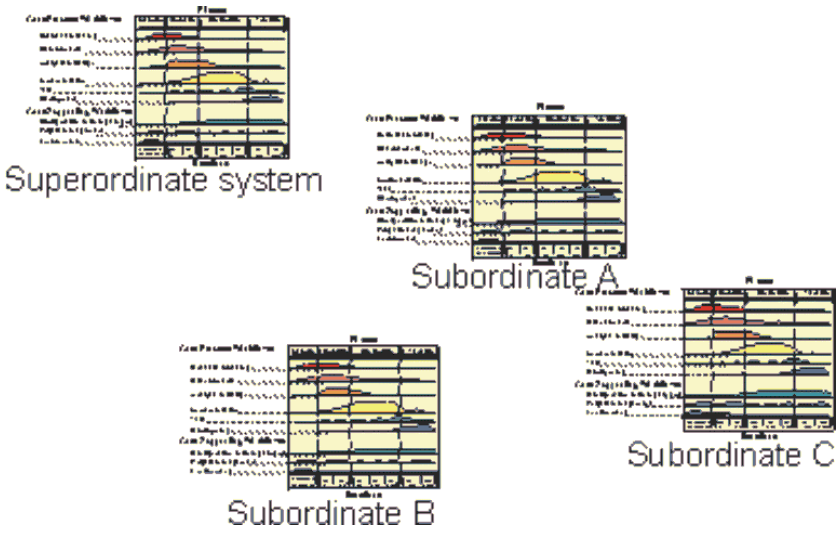

Figure 3. Chaque système, principal et subordonné, suit son propre cycle de vie.

Les cycles de vie ont, bien sûr, des dépendances. Les gérer correctement constitue l'un des défis du développement d'un ensemble de systèmes interconnectés. Elles peuvent être du type suivant :

- Les cycles de vie sont dépendants du temps. Le cycle de vie du système principal démarre le premier. Dès que le système principal a vécu au moins une itération et que les interfaces des systèmes subordonnés sont relativement stables, les cycles de vie des systèmes subordonnés peuvent démarrer. En fait, il se peut que vous ne sachiez même pas quels sont les systèmes subordonnés tant que vous n'avez pas passé au moins une itération sur le système principal.
- Le cycle de vie du système principal peut passer en mode maintenance dès que les interfaces des systèmes subordonnés sont stables. Ceci signifie qu'aucun développement actif n'est réalisé, sauf si des incidents se produisent, nécessitant des modifications au niveau des interfaces des systèmes subordonnés.
- Les interfaces des systèmes subordonnés appartiennent à ceux qui développent le système principal. Pour plus d'informations sur les interfaces, voir [3] et [5].
- Les classes qui mettent en oeuvre les interfaces de système subordonné sont détenues par ceux qui développement les systèmes subordonnés.

Flux de travaux et artefacts de développement système

Une hypothèse naturelle consiste à dire que le système principal, ainsi que les systèmes subordonnés, peuvent être développés avec le même ensemble d'artefacts et via les mêmes flux de travaux, typiques des systèmes non composites. Pour pouvoir montrer comment cela est possible, il nous faut d'abord présenter les artefacts et les flux de travaux. Dans le processus RUP (Rational Unified Process), cinq flux de travaux ont été introduits, comme le montre la figure 4. Il s'agit des flux de travaux suivants :

- Ingénierie métier : l'objectif est d'évaluer l'organisation dans laquelle le système va être utilisé, pour mieux comprendre les besoins et les problèmes devant être résolus par le système. Il en résulte un modèle de cas d'utilisation métier et un modèle objet métier. Ce flux de travail peut être considéré comme optionnel. Si l'organisation dans laquelle le système doit être utilisé est très simple, il se peut qu'aucune valeur ne soit ajoutée.
- Exigences : avec pour objectif de capturer et d'évaluer les exigences, en mettant l'accent sur l'ergonomie. Il en résulte un modèle de cas d'utilisation, avec des acteurs représentant des unités externes communiquant avec le système et des cas d'utilisation représentant des séquences de transactions, renvoyant des résultats ayant une valeur pour les acteurs.

- Analyse et conception : avec pour objectif l'analyse de l'environnement d'implémentation et de l'effet qu'il aura sur la construction du système. Il en résulte un modèle objet (le modèle de conception), incluant des réalisations de cas d'utilisation montrant comment les objets communiquent afin d'effectuer le flux des cas d'utilisation. Des définitions d'interface peuvent être incluses pour les classes et les sous-systèmes, spécifiant leurs responsabilités en termes d'opérations fournies. Ce modèle objet est également adapté à l'environnement d'implémentation en termes de mise en oeuvre du langage, de la distribution, etc. Il est parfois utile d'examiner les résultats d'analyse d'un modèle distinct, considéré ensuite comme le modèle d'analyse.

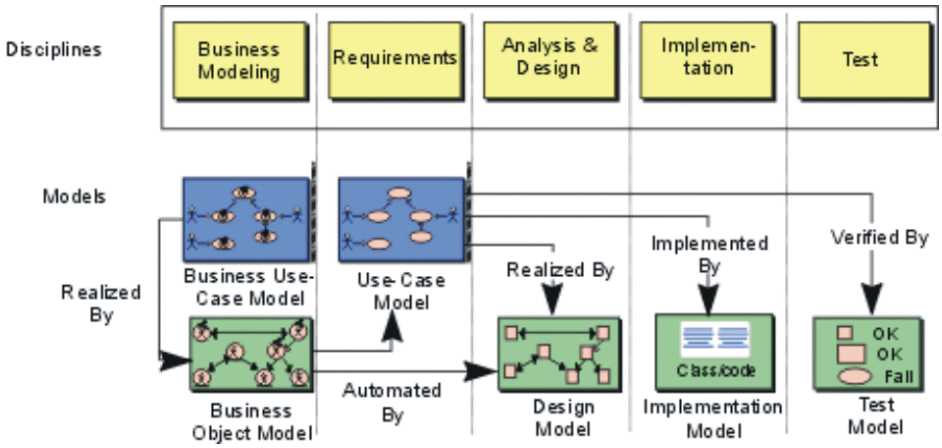

Figure 4. Chaque enchaînement d'activité principal est associé à un ensemble de modèles spécifiques.

- Implémentation : avec pour objectif de mettre en oeuvre le système dans l'environnement d'implémentation prescrit. Il en résulte un code source, des exécutables et des fichiers.
- Test : avec pour objectif de garantir que le système obtenu est celui attendu et qu'il n'y a pas d'erreur d'implémentation. Il en résulte un système certifié, prêt à être livré.

Développement d'un ensemble de systèmes interconnectés

Notre tâche consiste à définir comment les responsabilités d'un système peuvent être réparties sur plusieurs systèmes, chacun d'eux prenant en charge un sous-ensemble bien défini de ces responsabilités. Ceci signifie que le principal objectif est de définir les interfaces entre ces systèmes subordonnés. Une fois cette tâche accomplie, le reste du travail peut être effectué séparément pour chaque système subordonné, en fonction du principe "diviser-régner". Il s'agit là de tout ce que nous devons faire pour le système dans sa totalité, hormis le tester une fois son implémentation réalisée.

Critères de décomposition

Comment décider si vous devez décomposer votre système en un ensemble de systèmes interconnectés ? Un certain nombre de caractéristiques doivent être prises en compte :

- Pour un système de grande taille et complexe, vous pouvez fractionner le problème en plusieurs morceaux plus petits, dont la compréhension est plus simple lorsqu'ils sont pris individuellement.
- Etes-vous confrontés à des systèmes physiquement distincts ? C'est souvent le cas lors de l'utilisation d'une architecture ou de systèmes existants.
- La décomposition permet de définir des interfaces naturelles et étroites entre les diverses parties du système.

- Vous pouvez choisir de mettre en oeuvre une partie du système avec certains des principaux produits COTS. La décomposition permet également de clarifier comment vous avez l'intention d'utiliser le produit COTS.
- Le partitionnement permet d'obtenir le maximum d'une organisation de développement distribuée et de répartir clairement le travail entre plusieurs équipes dispersées géographiquement.

Les risques à prendre en considérations sont les suivants :

- Une utilisation excessive de la décomposition peut masquer le problème global pour tous les détails le concernant.
- En utilisant des systèmes ou des équipes physiquement séparés, vous encourez le risque de voir toute forme de réutilisation impossible et d'aboutir à un système hérité rigide.

Organisation

Afin de réduire les risques mentionnés ci-dessus, il est primordial qu'un groupe de personnes soit affecté au contrôle de l'effort de développement global. Ce groupe est souvent appelé équipe d'architecture et doit s'intéresser plus particulièrement aux problèmes majeurs suivants :

- Qu'une architecture globale soit définie et qu'elle soit suivie dans les systèmes subordonnés.
- Que l'accent soit raisonnablement mis sur la réutilisation et le partage de l'expérience entre les systèmes subordonnés.
- Qu'il soit clairement compris quels artefacts doivent être produits et quelles relations existent entre les artefacts des systèmes subordonnés et principaux.
- Qu'une stratégie de gestion des modifications efficace soit définie et appliquée par toutes les équipes.

L'équipe d'architecture peut, mais ce n'est pas toujours le cas, être responsable du développement du système principal. Pour une explication plus détaillée de l'organisation, voir [6].

Cycle de vie du système principal

Tout d'abord, vous pouvez choisir d'appliquer une **ingénierie métier** pour mieux comprendre le contexte du système. Cette opération constitue un ajout de valeur si :

- les développeurs ont besoin de mieux comprendre l'organisation,
- l'organisation est elle-même hétérogène dans sa façon de traiter ses affaires et sa terminologie et ses processus doivent être alignés, ou
- l'effort d'ingénierie logicielle est réalisé en accord avec celui de redéfinition des processus métier.

Voir aussi [6].

Cet effort doit aboutir à un modèle de cas d'utilisation métier et à un modèle objet métier. Vous pouvez également choisir d'appliquer une ingénierie métier limitée, en ne vous attachant qu'aux concepts clés du domaine métier et en les documentant dans un modèle objet métier. Ceci est souvent appelé modélisation du domaine.

Dès que vous avez amorcé le travail à l'aide d'un ensemble de modèles métier, vous devez commencer à recueillir les **exigences** de tout le système. Les exigences de modélisation sont les mêmes pour un ensemble de systèmes interconnectés que pour tout autre système. Un modèle de cas d'utilisation est un moyen très naturel pour exprimer les résultats, voir [7]. Le moyen le plus direct de considérer ce modèle de cas d'utilisation de principal est de présumer qu'il capture complètement les exigences de comportement du système. Cependant, c'est rarement le cas. Comme nous devons mettre en oeuvre le système avec d'autres systèmes, le système global est probablement un peu complexe. De ce fait, ce n'est pas une bonne idée de tenter d'être exhaustif à ce niveau. Aussi, un modèle de cas d'utilisation de principal donne généralement une image complète, mais simplifiée des exigences fonctionnelles du système. Il est donc inutile de trop entrer dans les détails à ce niveau, du fait que la modélisation détaillée s'effectue au sein de chacun des systèmes d'implémentation subordonnés. Il est souvent vrai que de

nombreuses exigences ne sont pas nécessairement visibles dans un cas d'utilisation de principal, divisant plusieurs sous-systèmes. De telles exigences peuvent être dites "locales" pour un sous-système.

Le but de l'**analyse et de la conception** est d'aboutir à une architecture solide du système qui de toute évidence, est d'importance vitale pour un ensemble de systèmes interconnectés. Les développeurs du système principal doivent parvenir à une robuste structure de systèmes subordonnés, alors qu'ils n'ont absolument pas besoin de s'inquiéter de leurs structures internes. De ce fait, nous allons modéliser une division du système en parties plus petites à l'aide de sous-systèmes. Afin d'obtenir le bon ensemble de sous-systèmes et une première idée de la répartition des responsabilités du système principal sur ces sous-systèmes, nous développons un modèle d'analyse. Les classes d'analyse doivent représenter les rôles joués par des éléments du système lorsque les cas d'utilisation de haut niveau sont exécutés. De ce fait, le modèle d'analyse fournit une image simplifiée de la structure d'objets complète, par analogie au modèle de cas d'utilisation de haut niveau.

Les classes d'analyse associées par fonction sont regroupées en sous-systèmes. Nous obtenons ainsi une structure de sous-systèmes qui est idéale, dans le sens où elle se base uniquement sur des critères fonctionnels. Par exemple, nous ne tenons pas compte des exigences de distribution. La présence de systèmes existants est souvent un facteur de forte influence. Les systèmes existants peuvent remplir certaines ou de nombreuses responsabilités définies dans le modèle d'analyse. L'existence de tels systèmes peut même entraîner un nouveau partage des responsabilités trouvées dans l'analyse de sorte que vous puissiez atteindre un niveau maximal de réutilisation des capacités existantes.

Le résultat de la conception peut être une structure de sous-systèmes, très différente de celle définie selon les critères fonctionnels au cours de l'analyse. Ainsi, nous aboutissons à une structure de sous-systèmes de conception qui sera mise en oeuvre par un système subordonné, voir la figure 5. Afin de pouvoir poursuivre la tâche de développement pour chaque système séparément, des interfaces sont définies pour chacun d'eux. En fait, la définition des interfaces représente l'activité la plus importante effectuée au niveau principal, car elles fournissent des règles de développement des systèmes subordonnés. Aucune classe de conception n'est définie, il vous suffit de définir les interfaces des sous-systèmes de conception.

Aucune **implémentation** n'est appliquée comme élément du cycle de vie du système principal, si ce n'est peut être qu'un travail de prototype afin d'explorer certains aspects techniques spécifiques du système.

La dernière activité est celle des **tests** qui dans ce cas sont des tests d'intégration une fois les différents systèmes subordonnés assemblés. Les tests consistent également à vérifier que le cas d'utilisation du principal est appliqué conformément à ses spécifications par les systèmes interconnectés fonctionnant en coopération.

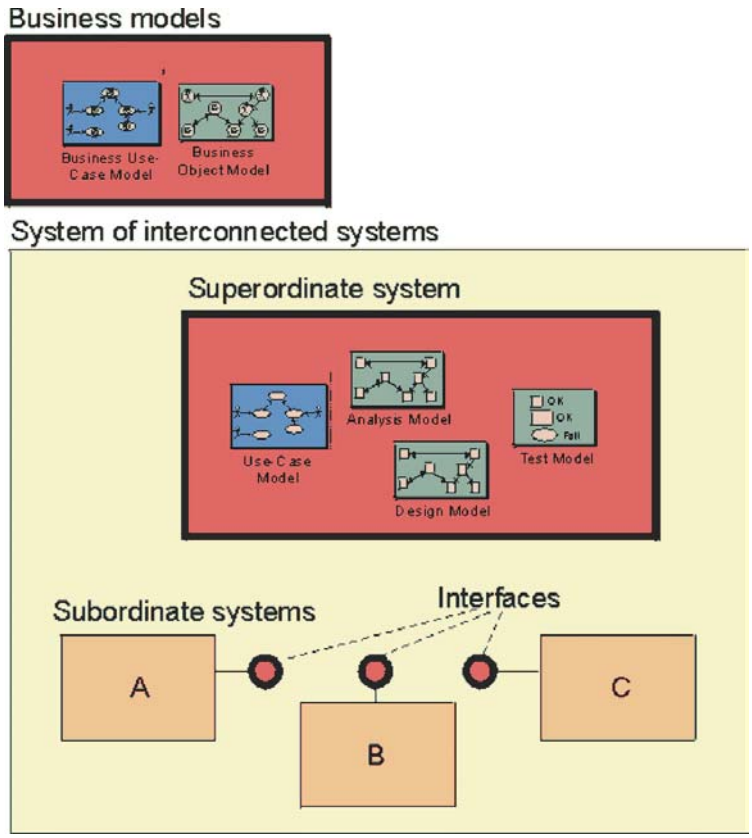

Figure 5. Le système principal est décrit par un ensemble de modèles dans lequel les sous-systèmes définis dans le modèle de conception de haut niveau sont implémentés par les systèmes subordonnés. Les interfaces avec les systèmes subordonnés appartiennent au système principal.

Les modèles de plan d'itération suivants permettent de vous montrer comment travailler avec le système principal : plan d'une itération au cours de la phase de création du cycle de vie du système principal et plan d'une itération au cours de la phase d'élaboration. Des diagrammes d'activité permettent de décrire les plans d'itération. Les états d'action mentionnés dans ces diagrammes correspondent aux détails des flux de travaux, tels qu'ils sont définis dans le processus RUP (Rational Unified Process).

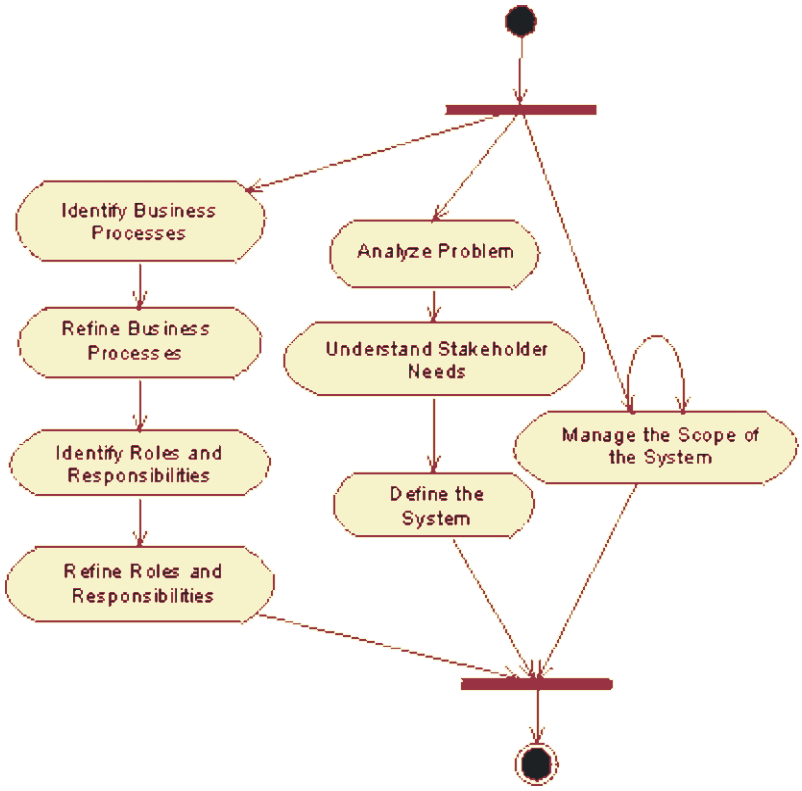

Figure 6. Diagramme d'activité décrivant un exemple de plan d'itération de création du système principal.

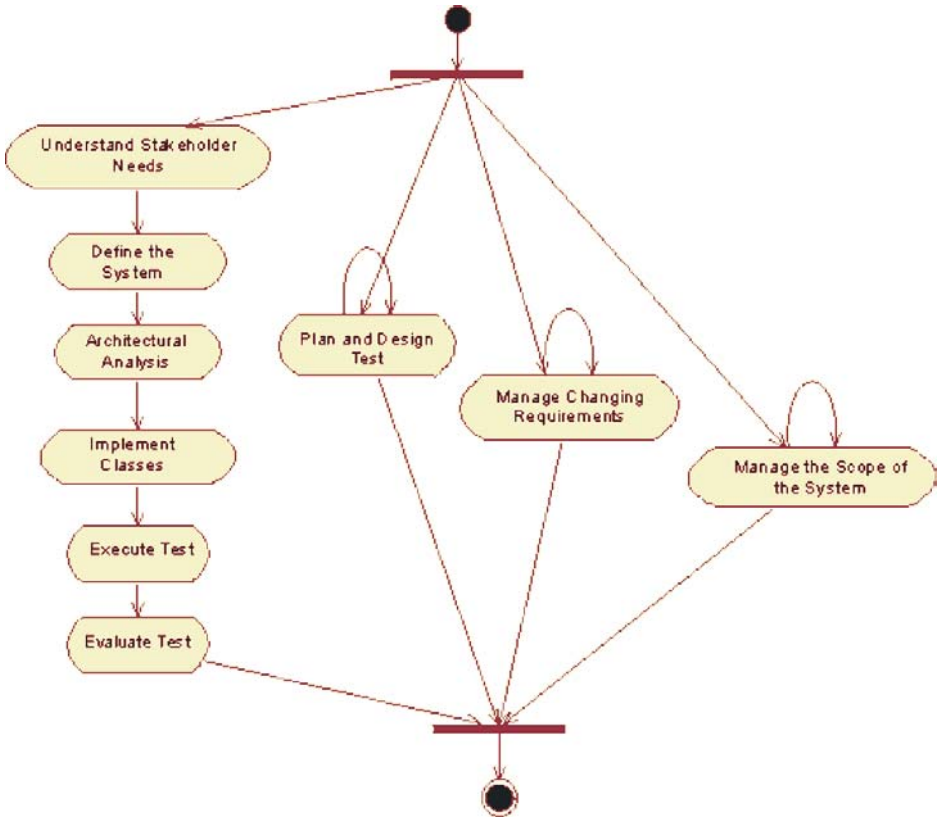

Figure 7. Diagramme d'activité décrivant un exemple de plan d'itération d'élaboration du système principal. L'état d'action "classes d'implémentation" figure ici, du fait que vous pouvez effectuer une implémentation limitée des prototypes afin d'explorer des aspects techniques du système.

Cycle de vie du système subordonné

Chaque système subordonné est développé de manière habituelle, comme une boîte noire tenant compte des autres systèmes avec lesquels il communique en tant qu'acteurs. Vous effectuez l'ensemble usuel d'activités et développez l'ensemble habituel de modèles, comme décrit ci-dessus, pour chacun des systèmes. Si au niveau du principal, la définition des modèles est totalement détaillée, vous obtenez une complète récursivité entre les modèles des différents niveaux, mais comme cela a été précédemment indiqué, ce cas est en pratique très rare.

Pour le système subordonné, vous devez effectuer le flux de travail des **exigences**. Les interfaces et les cas d'utilisation du système principal constituent votre principale entrée pour comprendre les limites du système subordonné et qui sont ses acteurs.

Lors de l'exécution de l'**analyse et de la conception** du système subordonné, les interfaces définies dans le système principal sont vos limites, associées aux cas d'utilisation de haut niveau.

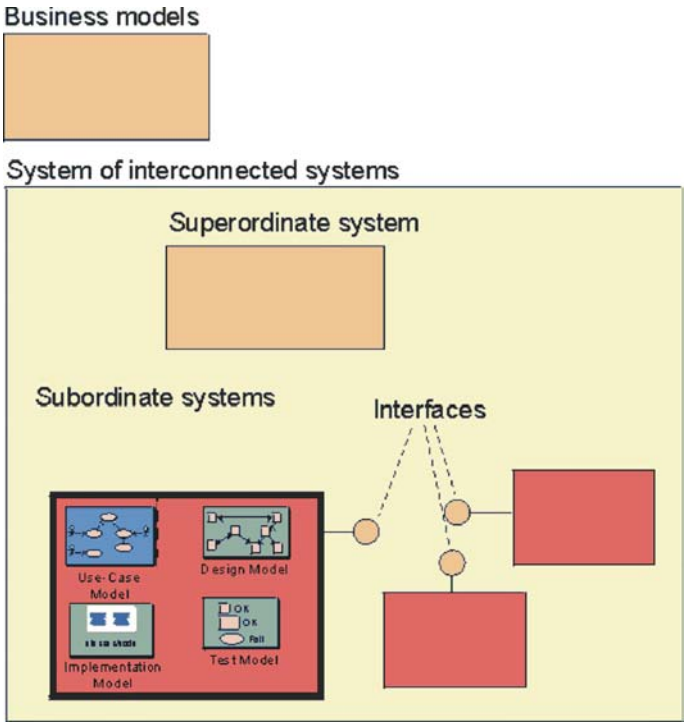

Figure 8. Les systèmes subordonnés sont décrits par leurs propres ensembles de modèles.

Les deux modèles suivants de plan d'itération basé sur le cycle de vie permettent de montrer comment utiliser le système subordonné.

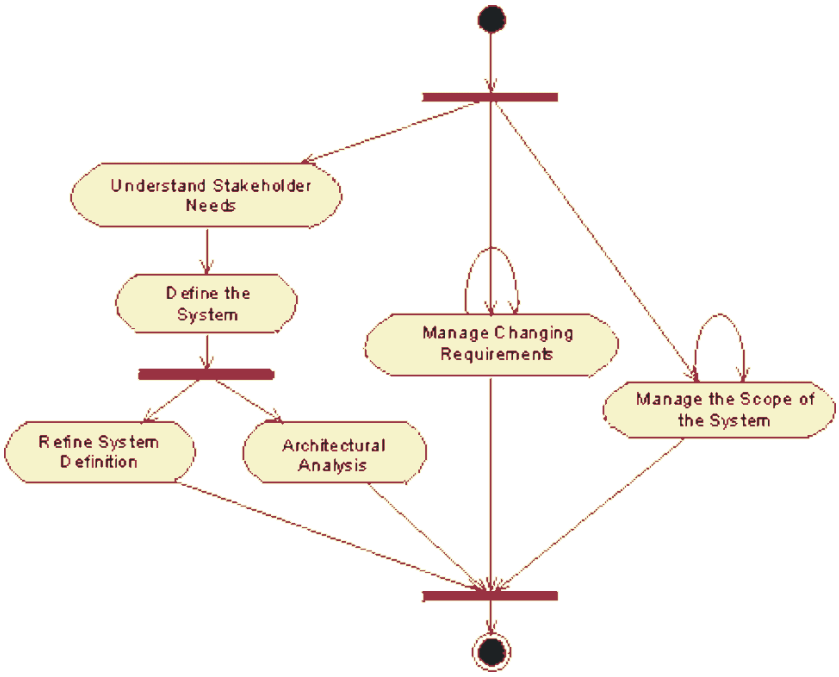

Figure 9. Modèle de plan d'itération de création du système subordonné. Cette itération est incomplète du fait qu'aucun exécutable n'est produit.

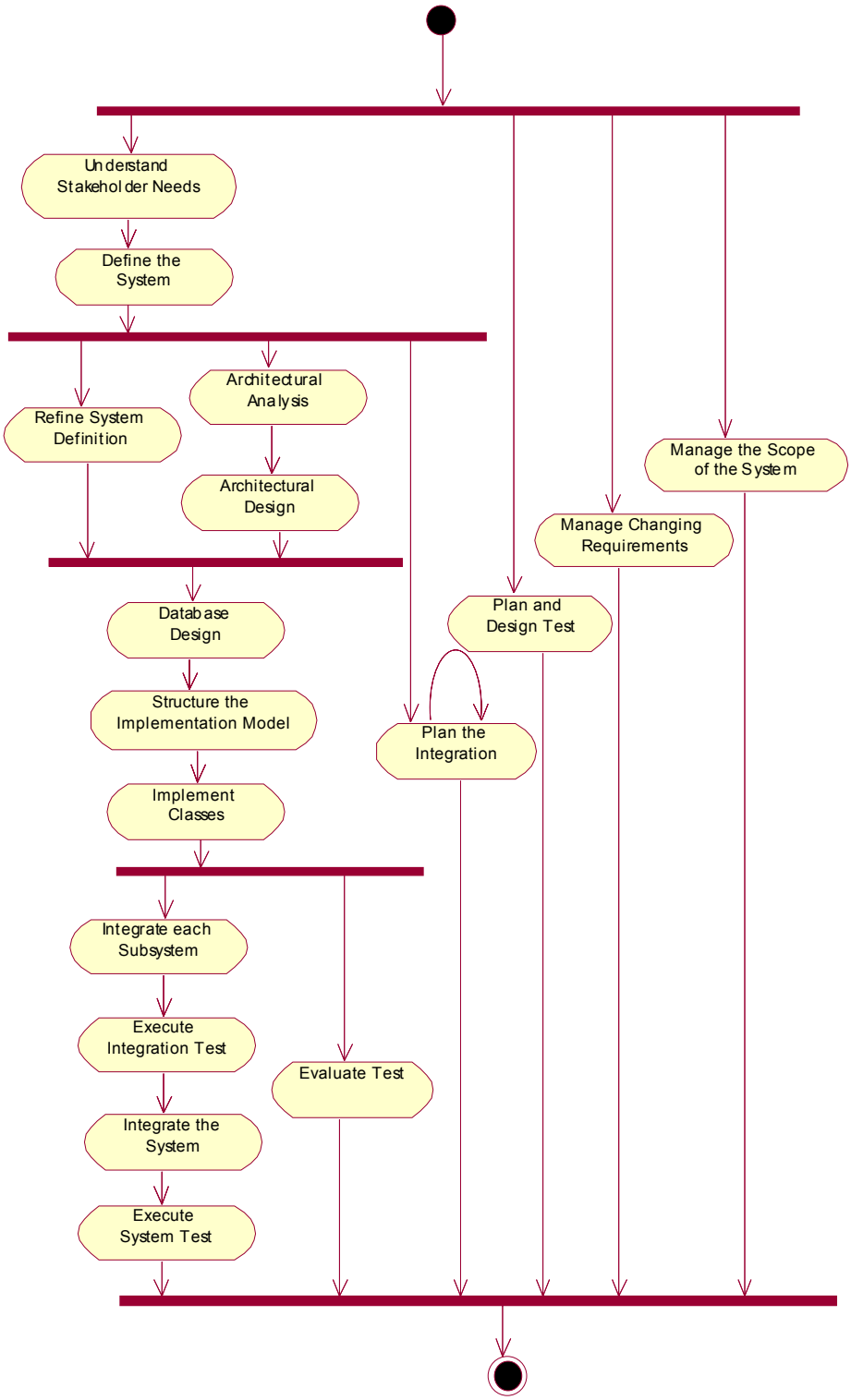

Figure 10. Modèle de plan d'itération d'élaboration du système subordonné. Dans l'élaboration, l'accent est mis sur la réalisation de la définition détaillée du système et de l'architecture.

Cas d'utilisation dans des ensembles de systèmes interconnectés

Un modèle de cas d'utilisation doit être généré pour chacun des systèmes, principal et subordonnés, dans vos ensembles de systèmes interconnectés. Ils sont dépendants de la façon suivante (voir aussi la figure 11) :

- Un cas d'utilisation de haut niveau dans le système principal est partagé (pas toujours, mais c'est souvent le cas) par les sous-systèmes. Chaque "part" devient un cas d'utilisation dans le modèle pour son système subordonné, voir la figure 11.
- Du point de vue d'un système subordonné, les autres systèmes subordonnés sont des acteurs du modèle de cas d'utilisation, voir la figure 12.

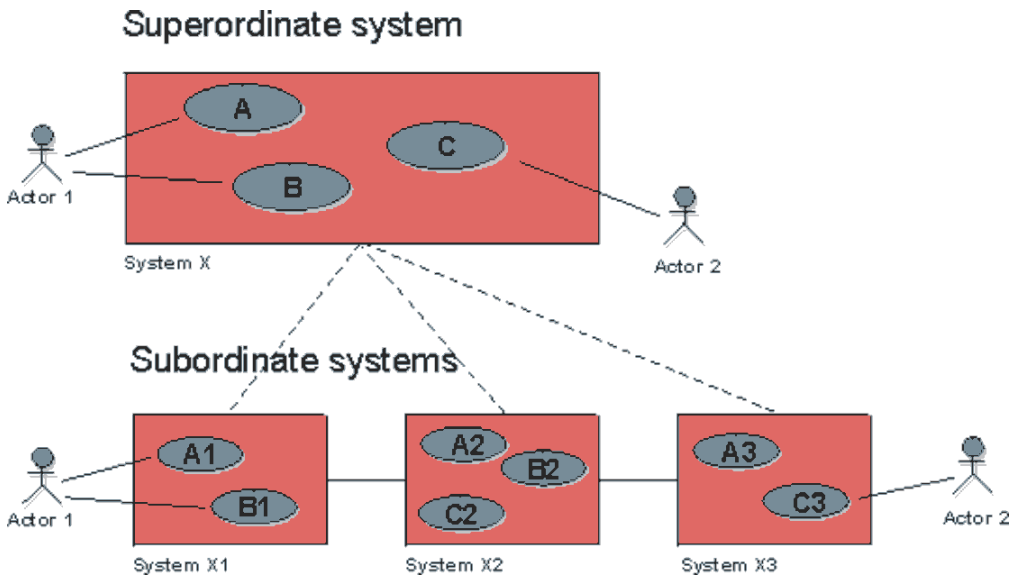

Figure 11. Relation entre le cas d'utilisation de haut niveau dans le système principal et les cas d'utilisation détaillés dans les systèmes subordonnés.

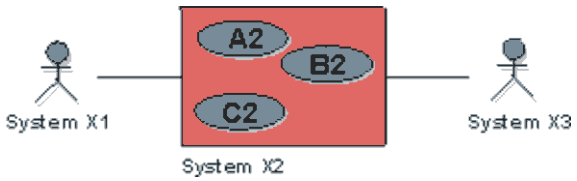

Figure 12. Dans le modèle de cas d'utilisation du système subordonné X2, les autres systèmes subordonnés X1 et X3 sont considérés comme des acteurs.

Les cas d'utilisation décrivant le système principal comportent des remarques spéciales. Comme dans un sens, vous allez décrire à nouveau toutes les exigences de chaque système subordonné, il est inutile de rentrer trop en détails dans ces cas d'utilisation. En règle générale, il est d'ordinaire suffisant de noter le chemin étape par étape jusqu'au flux d'événements des cas d'utilisation de haut niveau et inutile de le détailler sous forme de texte narratif.

Dans ce modèle de cas d'utilisation, vous ne devez pas utiliser les relations de cas d'utilisation (généralisation, extension, inclusion). En général, aucune valeur n'est ajoutée pour les raisons suivantes :

- Vous ne décrivez pas les cas d'utilisation de haut niveau en détails, aussi est-il inutile de vous inquiéter du texte apparaissant à plusieurs endroits.

- De toute façon, vous structurez les informations lorsque vous partagez les cas d'utilisation de haut niveau entre les systèmes subordonnés. Le mélange de ces faits avec d'autres mécanismes de structuration peut prêter à confusion.

Il existe cependant une importante exception qui consiste à trouver des composants réutilisables dans votre ensemble de systèmes interconnectés. La structuration du modèle de cas d'utilisation du principal pour trouver des cas d'utilisation génériques est une méthode puissante permettant de trouver des composants réutilisables. Pour plus de détails à ce sujet, voir [6].

Modèles de conception dans des ensembles de systèmes interconnectés

Chaque système de votre ensemble de systèmes interconnectés, principal et subordonnés, doit disposer de son propre modèle de conception. Les modèles de conception sont liés de la manière suivante :

- Les sous-systèmes du modèle de conception du système principal définissent les limites des systèmes subordonnés.
- Les opérations définies sur des sous-systèmes du système principal sont des entrées permettant de définir des interfaces avec les systèmes subordonnés.

Le modèle de conception du système principal est moins détaillé que les modèles de conception des subordonnés. Vous obtenez les éléments suivants :

- Sous-systèmes (brève description).
- Réalisations de cas d'utilisation, en termes de mode de collaboration des sous-systèmes. La façon habituelle de documenter ces réalisations de cas d'utilisation de haut niveau consiste à dessiner des diagrammes de séquence. C'est en produisant ces diagrammes que vous définissez le "partage" d'un cas d'utilisation de haut niveau sur les systèmes subordonnés, voir la figure 13.
- Opérations des sous-systèmes.
- Définitions des interfaces destinées aux sous-systèmes.

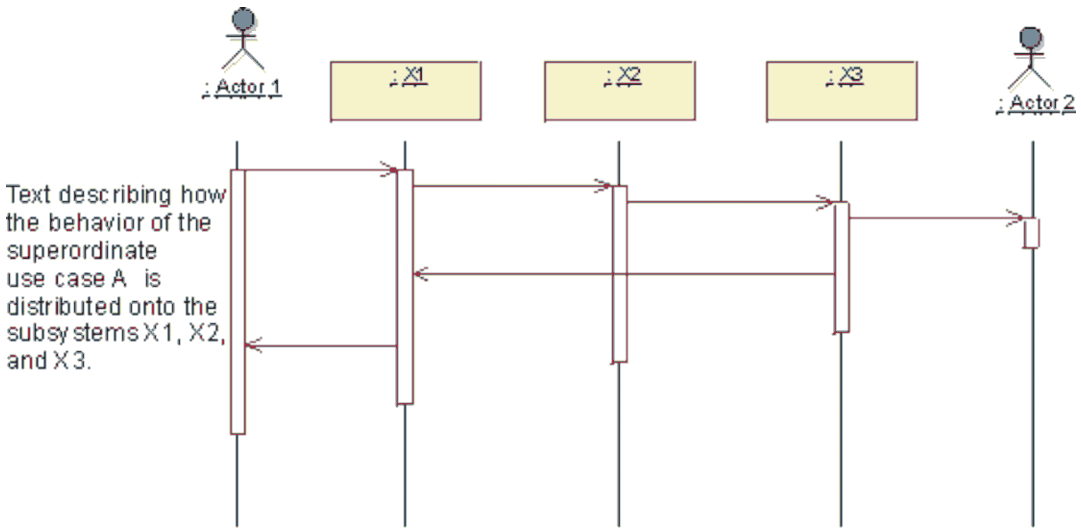

Figure 13. Diagramme de séquence pour la réalisation du cas d'utilisation A du principal.

Ensembles d'informations concernant les ensembles de systèmes interconnectés

Un effort important est engagé par la plupart des organisations pour comprendre comment gérer leurs artefacts et appréhender correctement leurs dépendances. A la section précédente, nous avons abordé en particulier la dépendance entre les modèles de cas d'utilisation du principal et des subordonnés et les modèles de conception. Certaines questions de dépendance générale doivent être également prises en considération.

Lorsqu'un système traverse une étape du cycle de vie, vous produisez des artefacts qui peuvent être organisés en ensembles d'informations [8], voir la figure 14. Ces ensembles sont organisés en fonction de l'évolution commune des artefacts.

- Vous pouvez personnaliser le contenu exact de chaque ensemble en fonction du type d'application que vous créez, mais les ensembles demeurent les mêmes.
- Vous devez comprendre les dépendances qui existent entre les ensembles, de façon à pouvoir gérer la traçabilité entre les artefacts de manière effective.

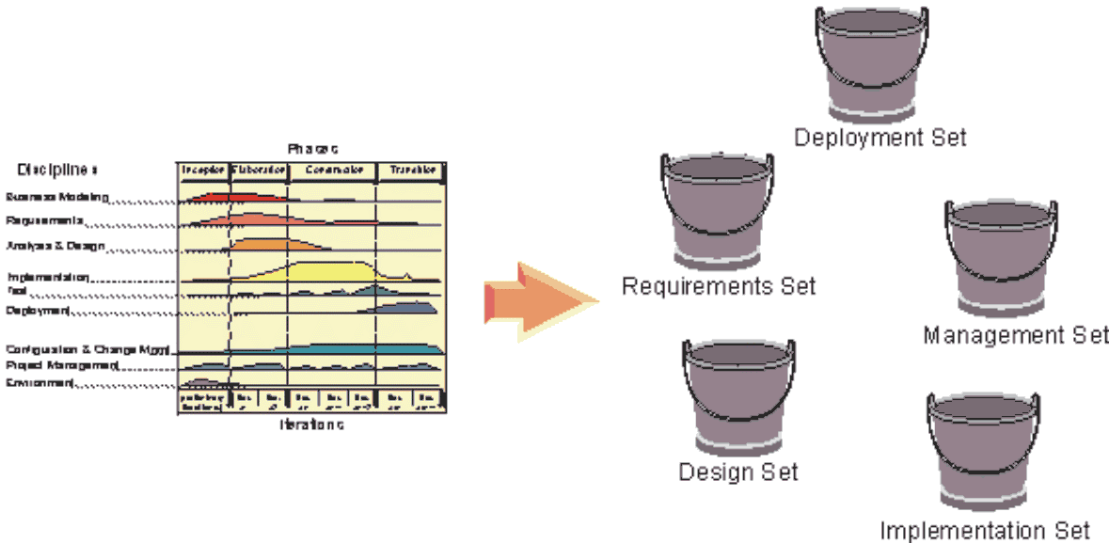

Figure 14. Le cycle de vie d'un système produit des ensembles d'informations.

Dans un ensemble de systèmes interconnectés, le système principal et chaque système subordonné produit son propre ensemble d'ensembles d'informations, voir la figure 15.

- Un ensemble d'informations subordonné possède des dépendances vis à vis de son ensemble d'informations principal correspondant.
- Le type de contenu peut différer dans les ensembles d'informations correspondants entre les divers systèmes subordonnés, car les types d'application peuvent être différents.
- Les ensembles d'informations subordonnés correspondants doivent être indépendants, mais ils constituent les mêmes interfaces de sous-systèmes définis dans le système principal.

L'effort placé pour gérer la traçabilité entre les artefacts du système principal et des systèmes subordonnés doit être maintenu au minimum. La priorité doit être mise sur la gestion de la traçabilité au sein du système.

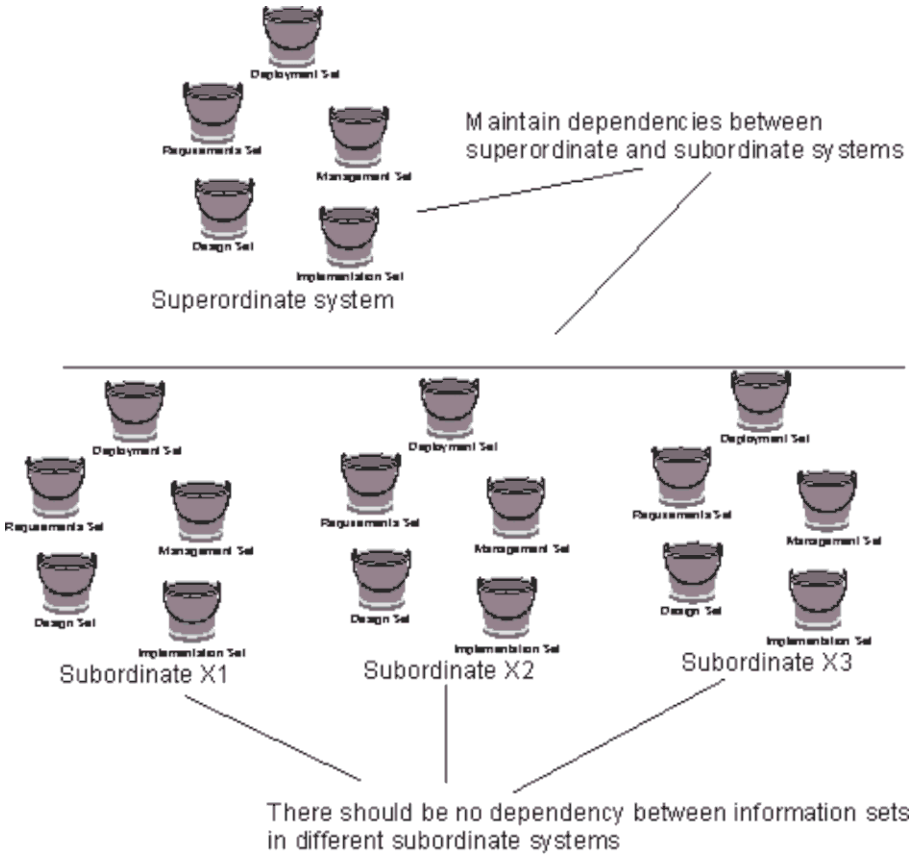

Figure 15. Chaque ensemble de systèmes interconnectés produit son propre ensemble d'ensembles d'informations.

Architecture d'ensembles de systèmes interconnectés

Chaque système de vos ensembles de systèmes interconnectés, principal et subordonné, doit avoir une architecture définie.

Pour le système principal, un document d'architecture doit aborder les points suivants :

- Principaux scénarios ou cas d'utilisation du système principal.
- Couches de l'ensemble de systèmes interconnectés.
- Comment gérer la réutilisation des systèmes subordonnés et que réutiliser ?
- Mécanismes clés et leur implémentation, mécanismes génériques suffisants pour être utilisés par tous les systèmes subordonnés. Par exemple, tous les systèmes subordonnés doivent utiliser des mécanismes communs pour la communication, le rapport des erreurs et la gestion des incidents, sinon le système principal ne se comporte pas comme un système homogène.

Pour les systèmes subordonnés, un document d'architecture doit être établi :

- Rôle du système subordonné au sein de l'ensemble de systèmes interconnectés.
- Principaux scénarios ou cas d'utilisation du système subordonné.
- Comment le système subordonné utilise-t-il la structure en couches définie pour l'ensemble de systèmes interconnectés ? En d'autres termes, vous devez définir comment le système subordonné remplit le rôle qui lui a été attribué dans l'architecture en couches du système principal.

- Quels mécanismes clés génériques sont utilisés et comment et quels mécanismes clés spécifiques à l'application sont ajoutés ?
- Comment la réutilisation s'applique-t-elle ? En particulier, quels sous-systèmes sont communs entre deux ou plusieurs systèmes subordonnés et quels mécanismes sont créés pour permettre aux systèmes subordonnés de communiquer ?

Relations entre systèmes

Nous avons vu que les activités de développement système habituelles s'appliquent également aux systèmes implémentés par des ensembles de systèmes interconnectés. Ceci est avantageux et signifie que vous n'avez pas besoin de gérer de tels systèmes de manière autre que celle utilisée pour les autres systèmes. Vous obtenez également une séparation simple du système principal de son implémentation sous forme des autres systèmes subordonnés. Chaque ensemble de systèmes interconnectés a son propre cycle de vie. Comme chaque système peut présenter des caractéristiques différentes, vous pouvez utiliser des variations du processus de développement pour les produire. En termes du processus RUP (Rational Unified Process) [2], vous disposez d'un cas de développement différent pour chaque système.

Dernière remarque sur l'indépendance des systèmes impliqués dans un ensemble de systèmes interconnectés :

Tout d'abord, examinons les systèmes subordonnés. Chaque système de ce type implémente un sous-système dans le modèle de conception du système principal. Les sous-systèmes dépendent des interfaces de chacun et pas explicitement les uns des autres, voir la figure 12. Ainsi, vous pouvez échanger un sous-système pour sa nouvelle version sans affecter les autres sous-systèmes, tant que le nouveau sous-système est conforme à la même interface. Vous obtenez exactement la même relation entre les systèmes subordonnés. Chaque système subordonné voit son environnement comme un ensemble d'interfaces. Ceci signifie que vous pouvez changer un système par un autre, tant que le nouveau système jour les mêmes rôles envers les autres systèmes ; c'est-à-dire tant qu'il peut être représenté avec le même ensemble d'interfaces. Les systèmes font référence aux interfaces de chacun d'entre eux, comme spécifié par les relations correspondantes entre les sous-systèmes et les interfaces du modèle de système principal.

Dans le modèle de cas d'utilisation d'un système subordonné, les interfaces des autres systèmes subordonnés avec lesquels il interagit sont représentés comme des acteurs. Nous pouvons dire qu'un système subordonné considèrent les interfaces d'un autre système, tel qu'offert par les acteurs correspondants, et de ce fait ne doit jamais référer directement à l'autre système, voir la figure 16. Notez que l'interface B apparaît à plusieurs endroits dans la figure 12, indiquant que la même interface est réellement référée par les sous-systèmes du système principal et par les systèmes subordonnés correspondants.

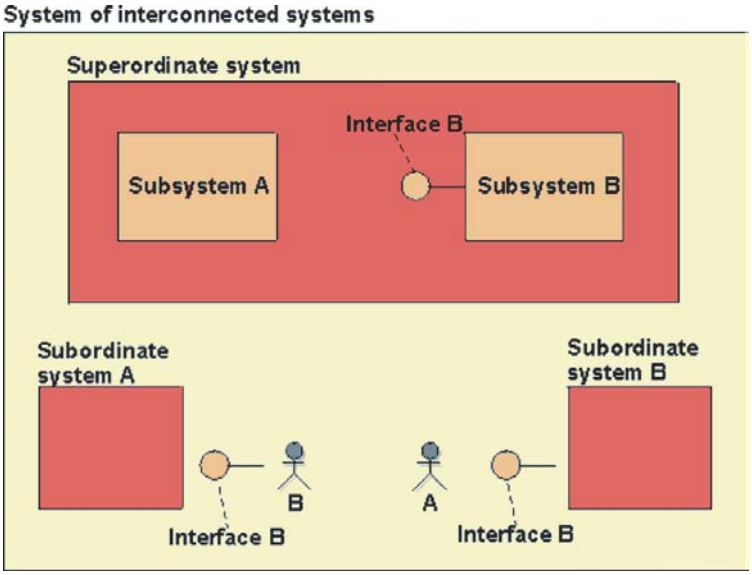

Figure 16. Les sous-ensembles de système principal dépendent les uns des autres, uniquement via leurs interfaces. Les systèmes subordonnés d'implémentation obtiennent de ce fait le même type d'indépendance. Dans le modèle de système principal, le sous-système B fournit l'interface B aux autres sous-systèmes. Le système B subordonné correspondant doit de ce fait fournir la même interface B aux autres systèmes subordonnés.

Quelle est la relation du système principal avec ses systèmes subordonnés ? Il est indépendant de ses systèmes d'implémentation dans le sens suivant : chaque système de ce type n'est qu'une implémentation de ce qui a été spécifié dans les modèles du système principal, il ne fait pas partie de sa spécification. Pour des raisons pratiques, vous devez définir des liens de traçabilité entre les systèmes à différents niveaux, afin d'effectuer le suivi des exigences et pour ce faire, la manière la plus "ordonnée" consiste à définir de tels liens uniquement entre les interfaces, voir la figure 11. En fait, nous pouvons même dire que les systèmes subordonnés ne sont rien de plus que des implémentations fournissant les interfaces définies dans les modèles de système principal.

Ceci est valable pour les systèmes qui sont plus que de simples exemples. Une interface ne spécifie rien d'autre que ce qui se passe sur un point d'interaction spécifique. Un système subordonné peut avoir des centaines d'interfaces et chacune de ces interfaces des dizaines d'opérations. Lier une entrée d'interface à une ou plusieurs sorties d'une autre interface est pratiquement impossible à faire dans une description d'interface. C'est pourquoi vous avez besoin de cas d'utilisation pour expliquer la sémantique du système subordonné.

Vous pouvez en conclure que chaque système impliqué dans une implémentation par un ensemble de systèmes interconnectés est dépendant des autres systèmes, mais principalement de leurs interfaces. Vous obtenez ainsi une très bonne plateforme de développement parallèle des systèmes subordonnés.

Zones d'application

L'architecture et les techniques de modélisation des ensembles de systèmes interconnectés peuvent être utilisées pour différents types de systèmes, tels que :

- systèmes répartis,
- systèmes très larges ou complexes,
- systèmes combinant plusieurs secteurs métiers,
- systèmes réutilisant d'autres systèmes,
- développement réparti d'un système.

La situation peut être également inverse : à partir d'un ensemble de systèmes existants, nous définissons un ensemble de systèmes interconnectés en les assemblant. En fait, dans certains cas, c'est de cette façon qu'un grand système évolue au cours des premières phases de son évolution. Vous réalisez que certains de vos systèmes peuvent être interconnectés, et ainsi vous créez un grand système qui ajoute plus de valeur que deux systèmes distincts.

En fait, pour tout système dans lequel il est possible de visualiser différentes parties du système en tant que systèmes à elles seules, il est préférable de le définir comme un ensemble de systèmes interconnectés. Même s'il s'agit d'un seul système aujourd'hui, il se peut que plus tard il s'avère nécessaire de le partager en plusieurs produits distincts, du fait du développement réparti, en raison de réutilisation ou des besoins des clients se limitant à n'en acheter que des parties, par exemple.

En conclusion, examinons plus en détails quelques cas dans lesquels l'architecture d'ensembles de systèmes interconnectés peut être utilisée. Pour chacun des exemples, nous allons montrer que le système en question doit être considéré *à la fois* comme un seul système *et* comme un ensemble de systèmes distincts, indiquant qu'il doit être traité comme un système principal, implémenté par un ensemble de systèmes interconnectés.

Grands systèmes

Le réseau téléphonique est probablement l'ensemble de systèmes interconnectés le plus grand du monde. C'est un exemple excellent lorsque plus de deux niveaux système sont nécessaires pour gérer la complexité. C'est également un exemple de cas dans lequel le niveau supérieur du principal appartient à un corps de normalisation et différentes entreprises concurrentes développent un ou plusieurs systèmes subordonnés qui doivent être conformes à cette norme. L'exemple traité ici est celui du réseau téléphonique mobile GSM (Global System of Mobile Telephony) qui montre les avantages de l'implémentation d'un grand système en tant qu'ensemble de systèmes interconnectés.

La fonctionnalité d'un très grand système associe généralement plusieurs secteurs métiers. Par exemple, la norme GSM couvre le système entier, de l'abonné appelant à l'abonné appelé. En d'autres termes, il inclut à la fois les comportements des téléphones mobiles et les noeuds réseau. Du fait que différentes parties du système sont des produits en elles-mêmes, achetés séparément, même par différents types de clients, elles doivent être traitées comme des systèmes. Par exemple, une entreprise qui développe des systèmes GSM complets vend les téléphones portables aux abonnés et les noeuds réseau aux opérateurs téléphoniques. C'est souvent une raison valable pour traiter différentes parties d'un système GSM comme plusieurs systèmes subordonnés. Une autre raison est qu'il serait trop long de développer un grand système complexe tel que GSM en tant qu'un seul système. Les différentes parties doivent être développées en parallèle par plusieurs équipes de développement.

D'autre part, comme la norme GSM couvre le système tout entier, il est utile de considérer également le système dans sa globalité, c'est à dire, le système principal. Ceci permet aux développeurs de comprendre le problème et de savoir comment les différentes parties sont liées entre elles.

Systèmes répartis

Pour les systèmes répartis sur plusieurs systèmes informatiques, l'architecture d'ensembles de systèmes interconnectés est parfaitement adaptée. Par définition, un système réparti est toujours constitué d'au moins deux parties. Des systèmes répartis nécessitant des interfaces bien définies, ces systèmes sont également très bien adaptés pour être *développés* de manière répartie, autrement dit par plusieurs équipes de développement autonomes, travaillant en parallèle. Les systèmes subordonnés d'un système réparti peuvent même être vendus en tant que produits. Ainsi, il est naturel de considérer un système réparti comme un ensemble de systèmes distincts.

Les exigences d'un système réparti couvrent généralement la fonctionnalité du système global et il arrive que les interfaces entre les différentes parties ne soient pas prédéfinies. De plus, si un domaine est nouveau pour les développeurs, ils doivent d'abord prendre en compte la fonctionnalité du système global, quelle que soit la manière selon laquelle il va être réparti. Il existe deux raisons très importantes pour le voir comme un seul système.

Réutilisation des systèmes existants

Il arrive que les grands systèmes réutilisent des systèmes existants. Le système hérité peut être alors décrit comme un système subordonné. Vous devez alors remanier un modèle de cas d'utilisation et peut-être un modèle d'analyse pour le système existant afin de comprendre comment il peut fonctionner dans le contexte plus grand du système principal. Ces modèles remaniés ne doivent pas nécessairement être complets. Ils doivent au minimum couvrir la fonctionnalité du système hérité ayant un impact direct sur la fonctionnalité du reste de l'ensemble de systèmes interconnectés, ou nécessitant une modification.

Utilisation de modules préfabriqués

Un système peut être l'intégration et la personnalisation de deux ou plusieurs modules préfabriqués. Les systèmes ERP (Enterprise Resource Planning) en sont un bon exemple. De nombreux systèmes ERP sont une composition de systèmes subordonnés, tels que MRP (Material Resource Planning), gestion des stocks et gestion de la chaîne d'approvisionnement, etc. Des compositions similaires sont disponibles dans d'autres secteurs, tels que les applications de paies ou de ressources humaines. Elles sont semblables à des systèmes préfabriqués que vous devez spécialiser et interconnecter avec d'autres modules standard afin d'obtenir un système complet. Pour comprendre ce que fait l'ensemble de modules, vous avez besoin du système principal. Cette situation est celle à laquelle de nombreux clients de la communauté financière sont confrontés aujourd'hui.

Résumé

Ce document introduit un schéma d'architecture pour les ensembles de systèmes interconnectés. Cette construction permet la récursivité non seulement au sein d'un modèle, mais elle considère également chaque sous-système comme un système à part entière et la récursivité a lieu entre tous les ensembles d'artefacts de chacun des systèmes. L'architecture introduite est utilisée pour les systèmes implémentés par plusieurs systèmes de communication. Chaque système impliqué est décrit par son propre ensemble de modèles, indépendamment des modèles des autres systèmes.

Les avantages que présente l'utilisation de cette technique sont évidents. Il est possible d'approcher des problèmes plutôt complexes et de les comprendre à l'aide de la technique "diviser pour mieux régner". Cependant, l'inconvénient réside dans le fait que vous risquez davantage de surcharge et de désynchronisation des plannings. Nous avons également vu des exemples dans lesquels les organisations rencontrent des difficultés à utiliser un cycle de vie itératif pour le système principal, engendrant ainsi le risque que des dommages soient entraînés à la fin du cycle de vie du système principal. Vous devez

également prendre garde au fait qu'une stratégie de réutilisation raisonnable et effective doit être appliquée afin d'éviter de développer un ensemble de systèmes hérités.

Les exemples cités montrent que l'architecture de modélisation d'ensembles de systèmes interconnectés est utile dans de nombreuses zones d'application différentes. En fait, vous pouvez utiliser l'architecture suggérée pour tout système dans lequel il est possible de visualiser les différentes parties en tant que systèmes en elles-mêmes.

Références

[1] Jacobson, I.; Palmkvist, K.; and Dyrhage, S., *Systems of Interconnected Systems*, ROAD, 2(1), 1995.

[2] *Rational Unified Process* version 5.1.

[3] Rumbaugh, J.; Booch, G.; Jacobson,I., *UML Reference Manual*, Addison Wesley Longman, 1999.

[4] Herbert A. Simon, *The Sciences of the Artificial*, MIT Press, 1981.

[5] Jacobson, I.; Bylund, S.; Jonsson, P., *Using Contracts and Use Cases to Build Plugable Architectures*, Journal of Object-Oriented Programming, mai/juin 1995.

[6] Jacobson, J.; Griss, M.; Jonsson, P., *Software Reuse – Architecture, Process and Organization for Business Success*, Addison Wesley Longman, 1997.

[7] Jacobson, I., *Use Cases in Large-Scale Systems*, ROAD, 1(6), 1995.

Rational
the software development company

Sièges :

Rational Software
18880 Homestead Road
Cupertino, CA 95014
Tél : (408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
Tél : (781) 676-2400

Appel gratuit : (800) 728-1212
Adresse électronique : info@rational.com
Site Web : www.rational.com
Sites internationaux : www.rational.com/worldwide