Université Nice SOPHIA ANTIPOLIS

Polytech'Nice-Sophia
Département Sciences Informatiques

Adaptation des interfaces à l'environnement

Towards a Multi-User Interaction Meta-Model

Tony BULTE

Le 11 novembre 2014

I - Présentation générale du sujet de recherche

Ce sujet de recherche s'intéresse principalement au travail en collaboration. Les principales questions auxquelles ce sujet de recherche tente de trouver des réponses sont comment organiser un groupe de personnes qui travaillent sur un objectif commun et comment organiser leur travail ?

Pour ce faire, les chercheurs travaillant sur ce sujet ont créé un méta-modèle suite à une analyse comparative de modèles de taches déjà existants.
Dans ce sujet de recherche, le méta-modèle généré est consacré au modèle de taches. Ce méta-modèle repose grandement sur des concepts comme l'ontologie et la taxinomie qui cherchent à modéliser les connaissances et à les hiérarchiser.
Pour pouvoir générer leur méta-model, les chercheurs ont dû sélectionner des modèles multi-utilisateurs qui supportent les interactions homme-machine et qui peuvent être intégrés à une méthodologie de développement. Ces travaux de recherche ont abouti au développement d'un éditeur recouvrant les principaux aspects du travail en collaboration. Ce méta-modèle est très clairement dans l'étage "modèle de taches" du modèle Cameleon.

Les principaux objectifs de ce sujet de recherche sont : fournir une analyse conceptuelle et méthodologique de quelques modèles de taches impliquant plusieurs utilisateurs. Créer un méta-modèle qui va reprendre les principaux concepts et les principales idées de modèles déjà existants. Le méta-modèle issu de cette recherche va donc être transversal à d'autres modèles. Enfin, l'objectif final de ce sujet de recherche est de créer une preuve de faisabilité de leur méta-modèle en développant un éditeur de modèles multi-utilisateurs.

II - Présentation de la solution

La solution finale proposée par ce sujet de recherche s'appelle "Multi-users interaction model".
Ce modèle est en fait un méta-modèle. Suite à une analyse comparative de différents modèles de taches déjà existants, ce meta-modèle reprend de nombreux concepts mis en évidence lors de cette analyse. Pour information, nous avons mis en annexe une description de chacun des modèles analysés par l'équipe de recherche.

Avant toute chose, la mise en place de ce meta-modèle nécessite d'identifier l'objectif que les utilisateurs vont devoir atteindre en collaborant. Cela va alors permettre d'identifier les taches, les rôles, les travails, et les ressources impliquées dans cette collaboration. Ceci va permettre aux utilisateurs de créer une conception complète de leur travail en amont.

Ce méta-modèle organise les taches en processus. Un processus est une ontologie de taches avec des relations de subsomption entre elles. Plus précisément, un processus est un graphe de taches avec des relations d'ordre entre elles pour connaitre l'ordre d'exécution de ces taches. Il y a différents processus, ceux dont les taches vont être exécutées séquentiellement et ceux dont les taches vont être exécutées parallèlement. Il est possible d'ajouter des conditions à un processus ou des itérations sur un processus ou sur ses taches.
Exemple de processus :

Organiser le décollage d'un avion

Figure 1 : Modelisation d'un processus

Avant de regrouper les taches en processus, il faut décomposer ces taches au niveau d'exécution le plus trivial possible. Ceci formant alors des arbres avec des opérateurs pour lier les taches qui sont au même niveau de décomposition. Un opérateur est une action cognitive et physique que l'utilisateur doit réaliser pour accomplir une tache.
Exemple de tache avec un opérateur :

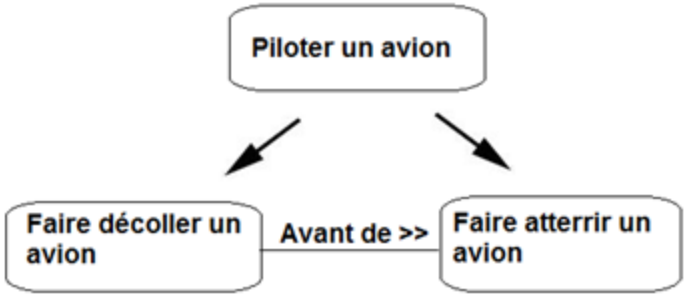

Figure 2 : Décomposition de taches

Une fois les tâches identifiées et organisées, il faut les distribuer les différents rôles qui ont les compétences pour les réaliser. La notion de compétence est importante. Cela

veut dire qu'avant d'attribuer un rôle à une personne, il va falloir connaître cette personne. La dimension humaine rentre alors en compte avec les problèmes de capacité, d'expérience ou encore de préférence.
Nous pouvons alors faire appel au concept de travail. Un travail va être attribué à chacune des ressources du système en fonction de leurs compétences et de leurs caractéristiques. La notion de travail regroupe les tâches, les devoirs et les responsabilités qui sont de même nature, c'est à dire qu'un travail correspond à un rôle. Par exemple, prenons le rôle de pilote d'avion. Les personnes qui ont ce rôle ont pour travail de conduire un avion d'un aéroport à un autre. Ils ont le devoir de piloter en étant en forme et ont le devoir de ne pas prendre de substances qui pourraient altérer leurs sens. Enfin, ils ont la responsabilité de satisfaire et de garder en vie leurs passagers.

Passons maintenant au concept d'unité organisationnelle. Cela représente en fait des personnes qui travaillent ensemble sur un même sujet. C'est un groupe de personnes qui ont un objectif commun.
Par exemple, un Ground handler est une unité organisationnelle :

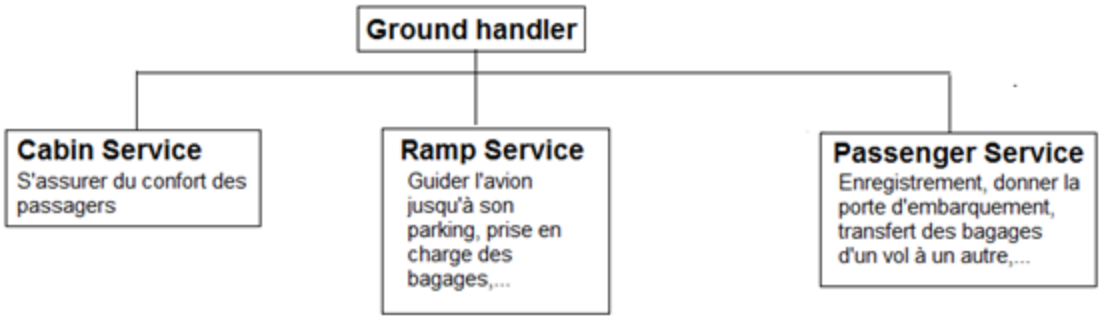

Figure 3 : Unité organisationnelle

Ce concept est intéressant dans le sens où dans un groupe de travail, il va y avoir un certain nombre de besoins et notamment celui de partager des objets de travail. Par exemple, lorsque deux pilotes d'avion ont pour tâche d'atterrir sur une seule piste au même moment, les pilotes vont devoir se partager la piste d'atterrissage pour tous les deux pouvoir réaliser leur tâche. Le problème qui découle de cet exemple est la coordination. Les deux avions ne vont pas pouvoir atterrir au même moment sur la piste car elle ne peut accueillir qu'un avion à la fois. Ainsi la question est : comment ces deux ressources vont coordonner leurs actions pour pouvoir réaliser tous les deux leur tâche ?
Ces objets, peuvent aussi bien être matériels (un avion, une piste d'atterrissage) que immatériels (de l'électricité, des ondes radio).

Le dernier concept comportant utilisé par ce meta-modèle est l'agenda. Un agenda regroupe des tâches assignées à un type d'utilisateur (rôle). L'agenda va permettre de faire face aux aspects coopératifs lors d'un travail en groupe. Grace à cet agenda, on va assigner des taches à un certain type d'utilisateur. De plus, cet agenda est accessible par les personnes du groupe de travail. Cela va ainsi permettre de suivre la collaboration des ressources, de notifier les ressources à propos des taches qui peuvent leur être associées. Un agenda va aussi permettre aux ressources de prendre ou de déléguer une tache. Enfin, ça va permettre aux ressources de voir comment les taches sont développées et de savoir qui est en charge de quelles taches.

Voici ci-dessous un diagramme représentant les concepts vu précédemment :

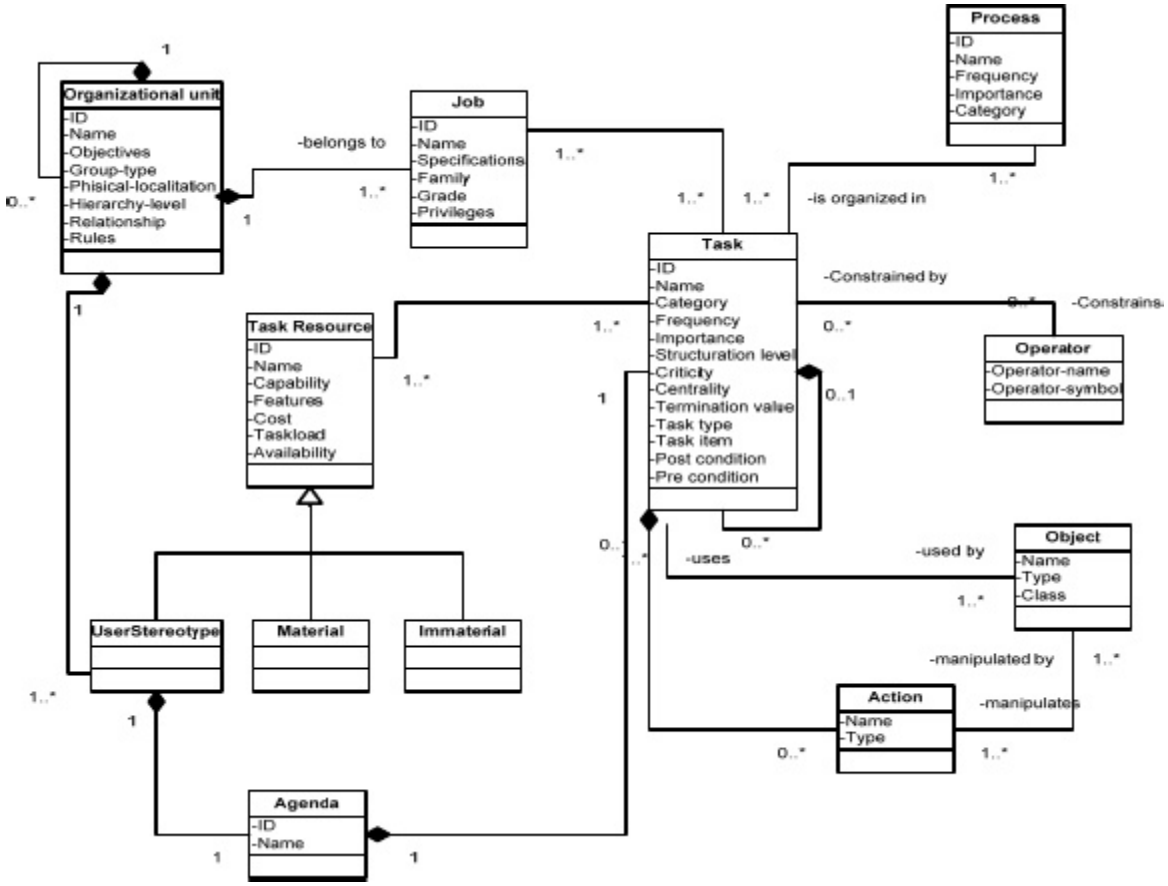

Figure 4 : Multi-User Interaction Meta-Model

Pour illustrer cette théorie, reprenons notre exemple et donnons-nous comme mission de faire voyager un avion d'un aéroport A jusqu'à un aéroport B.

Pour mettre en oeuvre ce voyage, il est nécessaire de connaitre les taches, les rôles et les ressources (objets) impliqués.

Commençons par définir les taches et leurs sous-taches avec un ordre d'exécution :

Piloter un avion	1)Faire décoller un avion 2)Faire atterrir un avion
Gérer la coordination pour l'atterrissage et le décollage des avions sur un aéroport	1)Autoriser un avion à décoller 2)Autoriser un avion à atterrir 3)Attribuer un stand à un avion qui vient d'atterrir
Prendre en charge un avion au sol	1)Faire embarquer les passager et leurs bagages 2)Guider un avion jusqu'à sa piste de décollage 3)Guider un avion jusqu'à son stand 4)Débarquer les passager et leurs bagages

	5)Faire le pleins de kérosène pour le vol suivant

Figure 5 : Taches et leurs sous-taches

Maintenant, ordonnançons ces taches :
1)Faire embarquer les passager et leurs bagages
2)Guider un avion jusqu'à sa piste de décollage
3)Autoriser un avion a décoller du point A
4)Faire décoller un avion à partir du point A
5)Autoriser un avion à atterrir à un point B
6)Faire atterrir un avion à un point B
7)Attribuer un stand a un avion qui vient d'atterrir
8)Guider un avion jusqu'a son stand
9)Débarquer les passager et leurs bagages
10)Faire le pleins de kérosène pour le vol suivant

A partir de ces taches, nous pouvons définir les rôles :

-Le pilote : Son travail va être de piloter un avion
-La tour de contrôle : Son travail va être de gérer les arrivées et les départs d'avions. La tour de contrôle est une unité organisationnelle (C'est un groupe de personnes)
-Le Ground Handler : Son travail va être de s'occuper des avions au sol. Le ground handler est une unité organisationnelle (C'est un groupe de personnes)

Nous pouvons maintenant associer ces rôles aux taches :

Taches	Rôle associé
1)Faire embarquer les passager et leurs bagages	Ground handler
2)Guider un avion jusqu'à sa piste de décollage	Ground handler

3)Autoriser un avion a décoller du point A	Tour de contrôle
4)Faire décoller un avion à partir du point A	Pilote
5)Autoriser un avion à atterrir à un point B	Tour de contrôle
6)Faire atterrir un avion à un point B	Pilote
Remarque : Apres l'exécution des sous-taches 4 et 6, la tache "Piloter un avion" est terminée.	
7)Attribuer un stand a un avion qui vient d'atterrir	Tour de contrôle
Remarque : Apres l'execution des sous-taches 3, 5 et 7, la tache "Gerer la coordination pour l'atterrissage et le décolage des avions sur un aéroport" est terminée.	
8)Guider un avion jusqu'à son stand	Ground handler
9)Débarquer les passager et leurs bagages	Ground handler
10)Faire le pleins de kérosène pour le vol suivant	Ground handler
Remarque : A ce stade, nous avons atteint notre objectif de départ qui était de faire partir un avion d'un point A à un point B.	

Figure 6 : Processus et rôles associés aux taches du processus

Pour cet exemple, admettons que nous connaissons déjà les compétences des personnes qui vont collaborer ensemble. Nous leurs attribuons alors les rôles et travails adéquates.

Remarquons que le méta-modèle utilisé peut permettre d'attribuer plusieurs rôles à une personne. Pour notre exemple nous n'avons pas réellement besoin d'attribuer plusieurs rôles à une personne.

Dans notre ordonnancement de taches, nous avons des contraintes temporelles. Un pilote ne peut pas faire décoller son avion avant d'avoir reçu l'autorisation de la tour de contrôle.

Dans notre exemple, il est difficile de mettre en place un agenda. L'idée serait que le pilote informe la tour de contrôle lorsqu'il veut décoller ou atterrir. Apres cela, la tour de contrôle serait en mesure d'informer le ground handler si un avion se trouve au sol. Ainsi, chacun des acteurs serait informé de l'avancement de chaque tache.

Enfin, nous pouvons affirmer que notre voyage en avion est un succès. En effet, grâce au méta-modèle d'interaction multi-utilisateur, nous avons su déterminer les différents besoin nécessaires à la réalisation de cet objectif :

-La définition des taches
-La définition du processus de taches
-Les objets (avion, radio, piste d'atterrissage,...)
-Les ressources (Pilote,...)
-Les unités organisationnelles (Tour de contrôle, Ground handler)
-Les travails/rôles
-L'agenda
-les différents objectifs (Les cases en vert dans la figure 3)
-Les conditions/règles (un avion ne peut pas décoller tant qu'il n'a pas reçu l'autorisation,...)

III - Conclusion

Sachant qu'il y a de nombreux modèles de taches avec différentes approches, il est très difficile d'effectuer une analyse comparative entre eux. Dans cet article de recherche, pour pouvoir générer leur méta-model, ils ont du sélectionner les modèles qu'ils pouvaient analyser théoriquement, qui supportaient les interactions homme-machines et qui pouvaient être intégré à une méthodologie de développement. L'objectif final de cette recherche était d'analyser les modèles afin d'en tirer les principales caractéristiques pour concevoir un travail dans la collaboration. Et de mettre en œuvre cette analyse en créant un méta-model à partir des modèles analysés.

Le meta-modèle permet donc de savoir comment structurer les taches, qui va les réaliser, quel est l'ordre entre ces taches, et comment elles sont assignées et suivies. Celui-ci recouvre les principaux aspects du travail d'équipe. Cela inclus donc le concept de processus de tache, de collaboration, d'actions, de ressources, de rôles, d'agenda, d'objectifs et de règles/conditions.

Me concernant, je pense que ce modèle est très complet, il rassemble tout ce dont un travail d'équipe a besoin, c'est à dire de l'organisation (tache, rôles), de la communication dans l'équipe (agenda), et le système d'objectif me permet d'envisager une collaboration agile avec la mise en place de Sprint (Je pense que ce modèle pourrait se coupler avec des techniques comme Scrum ou kanban).
Je pense que les principaux avantages sont que les utilisateurs peuvent travailler en collaboration avec un objectif commun et avec des rôles bien définis. Cela permet d'organiser les personnes et de leur donner un scope.
De plus, les informations circulent bien entre les utilisateurs. (Tout le monde est au courant de ce qui se passe, de qui s'occupe de quoi,...). Mon premier mois en entreprise en tant qu'alternant m'a appris qu'avoir une bonne communication est importante. Sinon, des confusions et des troubles peuvent se créer au sein d'une même équipe.
Enfin, l'avantage avec ce genre de modèle très théorique est que l'on peut les adapter très facilement à des contextes d'usages plus spécifiques. On peut ajouter ou retirer des éléments de modélisation sans trop de soucis pour qu'il convienne mieux à notre cas d'utilisation.

Ensuite, concernant les inconvénients, il est très difficile de prédire ce dont on va avoir besoin pour une tache. Il va toujours y avoir des imprévus en termes de besoin. Ce modèle reste très théorique et je pense pas facile à mettre en place.

Si jamais il me venait à utiliser ce meta-modèle, je pense qu'il faudrait obligatoirement réserver un rôle pour la gestion et la mise en place de ce modèle. Il faudrait une sorte de Scrum master qui encadrera les membres de l'équipe pour travailler en appliquant le modèle correctement. Cela fait donc une ressource en moins sur la collaboration. Mais la question est est-ce qu'il vaut mieux supprimer une ressource mais avoir une bonne organisation ou est-ce qu'il vaut mieux avoir une organisation approximative mais avoir toute les ressources nécessaires ? Je pense que la réponse à cette question va énormément dépendre des projets et des équipes elles-mêmes.

Un des principaux inconvénients, qui s'applique globalement à tous les patterns, est qu'un modèle ne peut pas s'appliquer à tous les types d'utilisations possibles.

Sources :

http://www.w3.org/2005/Incubator/model-based-ui/wiki/Task_Meta_Models#AMBOSS_Task_Meta-Model

http://www.academia.edu/2757297/Towards_a_multi-user_interaction_meta-model

Annexe : Description des différents modèles utilisés pour l'analyse comparative du sujet de recherche

Ci-dessous se trouve la description des de chacun des modèle analysés par l'équipe de recherche dans le but de créer leur propre méta-modèle. Ces descriptions sont volontairement très générales car le cœur du sujet de recherche est le méta-modèle généré.

Modèles de taches :

-Groupware Task Analysis

Ce model met en valeur la diversité des taches dans un environnement multi-utilisateur. Avec ce modèle, tous les utilisateurs ont un objectif commun mais vont se voir attribuer des rôles des sous-objectifs différents.
Ce model valorise tout d'abord les rôles et sous-rôles des utilisateurs avec en prime une relation de responsabilité entre les rôles et les taches.
Ensuite, il met l'accent sur le travail, et surtout sur la création d'objectifs, la décomposition des taches et les évènements qui déclenchent ces taches. Il faut garder en tête qu'une tache est définie pour être réalisée par un certain type de rôle. Par exemple, si nous considérons la tâche "trouver des bêta-testeurs pour notre projet", il serait pertinent d'affecter cette tache a quelqu'un qui a le rôle de "commercial" et qui a les bonnes compétences.

Enfin, pour réaliser une tache, un utilisateur, ou un groupe d'utilisateurs vont avoir besoin "d'objets", ou plus précisément d'outils
Pour conclure, les principes fondamentaux de ce model sont :

-Les utilisateurs jouent des rôles, et ces rôles sont associés à des taches.

-Les utilisateurs exécutent des actions dans le but de réaliser une tache

-Les utilisateurs vont manipuler des objets qui sont utilisé par les taches elles-mêmes.

-Les taches sont déclenchées pas des évènements.

-Task Knowledge Structure

Ce modèle est plus complexe que le précédent car il prend en compte l'aspect "humain" et l'aspect de "collaboration".
Ce modèle est une représentation conceptuelle des connaissances qu'une personne a à propos de taches spécifiques.

Ce model s'intéresse aux rôles et aux responsabilités que l'on peut définir en fonction des différentes taches prédéfinies. Ici, une personne n'a pas un rôle fixe, une personne peut porter plusieurs "casquettes". Ceci autorise une personne à exécuter des taches similaires sous différents rôles.

Le TKS va aussi permettre de définir une structure d'objectifs, avec des objectifs et des sous-objectifs. On applique ici le principe de "diviser pour mieux régner". Ainsi,
au lieu de réaliser une tache entière d'un seul coup, on va réaliser plusieurs petites taches qui vont nous mener à la réalisation complète d'une tache. C'est en quelques sorte une "roadmap" (=procédure) qui va définir le chemin à prendre pour partir d'un point A et arriver à un point B.

-Fundamental Knowledge Structures

Ceci est la clé de voûte des interactions multi-utilisateurs. Pour une bonne collaboration, il faut prendre en compte les différentes sortes de savoirs des utilisateurs. D'abord, il y a la connaissance générale de ce qui fait qu'une collaboration est efficace. C'est à dire que les utilisateurs ont des connaissances complémentaires et arrivent à comprendre et accepter les connaissances d'autrui.
Ensuite, il y a la question de comment les utilisateurs vont collaborer pour réaliser une tâche, et quelles vont être les contributions de chacun à la réalisation de cette tâche.
Enfin, il y a la connaissance de l'autre. Est-ce qu'un utilisateur est capable de dire quelles sont les connaissances d'un autre utilisateur, et est-ce qu'il arrive à comprendre le domaine d'application de ses connaissances et le contexte d'application de ces connaissances.

Ainsi, le "fundamental knowledge structures" va permettre d'engendrer un ensemble de prérequis pour la collaboration sur les différentes taches.

-Concur Taks Tree

Ce modèle fait intervenir 5 concepts qui sont :

- -Les taches
- -Les objets
- -Les Actions
- -Les operateurs
- -Les rôles

Ces concepts sont reliés tel qu'un rôle est responsable d'une tache, une tache est contrainte par un opérateur, une tache est composée d'actions qui utilisent des objets.

Ce modèle va notamment permettre de décrire la coopération des taches en elles. Comment est-ce qu'elles vont communiquer ou interférer entres elles. Notons que chaque tache dite "coopérative", c'est à dire que plusieurs personnes sont nécessaires à sa réalisation, va pour avoir un modèle pour son aspect coopératif et d'autres modèles pour chacun des rôles impliqués à sa réalisation.

Ensuite, il faut passer par l'étape de décomposition des taches jusqu'à ce qu'elles soient d'une compréhension et d'une implémentation triviale. Cela va alors mener à la conception d'un ou plusieurs graphes de taches.

Ensuite, pour chacune de ces taches basiques, on spécifie les objets et actions associés. Pour pouvoir être compris et présenté aux utilisateurs, les objets doivent être des objets perceptibles. C'est à dire que l'utilisateur doit pouvoir associer cet objet à quelque chose qu'il connait déjà.

Enfin, pour que l'utilisateur sache comment interagir avec ces objets et pour savoir quelles sont les conséquences de ces interactions, il faut spécifier les actions "input" et les actions "output" associés aux objets.

-Task Object-oriented Description

La particularité de ce modèle est qu'il voit les taches comme des méthodes orientées objet. La première étape pour appliquer ce modèle est de décomposer hiérarchiquement les taches. Cela va permettre de créer un arbre de tache avec des taches "nœuds" qui vont potentiellement être des taches complexes car elles seront composées d'autres taches. Et il va y avoir des taches "feuilles" qui seront les taches les plus basiques avec un niveau de complexité quasiment nul. Chacune de ces taches est considérée comme une instance d'une classe identifiée par un nom et caractérisé par un objectif, un type (automatique, coopérative, ...), son niveau dans l'arbre des taches et son nombre de sous-taches.
Ensuite, chaque tâche est associée à une "structure de contrôle de tache" qui est une structure complexe et complète permettant de relier les taches entre elles.
Chaque tâche est donc reliée à :

-Un déclencheur (trigger). Il y a quatre types d'évènements : les évènements formels, informels, qui proviennent de l'extérieur ou qui proviennent de l'intérieur du système.

-Des conditions. Une fois ces conditions remplies, la tache peut alors être exécutée (exemple : Ma tâche ne peut être exécutée qu'après celle de mon collègue).

-Des ressources. Ce sont les objets nécessaires à la réalisation de la tâche. (Exemple : un ordinateur pour pouvoir remplir la tâche "écrire un rapport").

-Des informations d'entrée. Décrivent ce qui est requis pour effectuer et/ou initialiser une tache, les règles permettant de s'assurer que toutes les conditions sont réunies pour effectuer la tâche.

-Des informations de sortie. Ces informations décrivent ce que la tache produit et décrivent aussi les règles de synchronisation avec les autres taches.

-Des réactions physiques et cognitives résultantes de l'exécution de la tâche.

Pour conclure sur ce modèle, il considère les taches comme des objets et il les ordonne de façon à avoir une hiérarchie de taches toutes associées à une structure de contrôle.

-Diane Meta-model

Ce modèle est utilisé lors de la phase d'analyse des besoins des utilisateurs. Il va permettre de fournir une description des procédures standards à utiliser lors de la réalisation d'une tache. Ces procédures peuvent être par exemple une action "quitter" ou encore une action "annuler". Ces procédures ont besoin d'un évènement en entrée comme par exemple "l'utilisateur a appuyé sur le bouton annuler", cela va alors déclencher une action exécutée par le système.

Ce modèle va ainsi dire si une tache va être exécutée par un utilisateur, par le système ou par les deux.

-Hierarchical Task Analysis

Ce modèle va permettre de modéliser des taches complexes. Il se base principalement sur des interviews ou des observations d'utilisateurs et sur l'analyse de documentations.

Dans ce modèle, les taches vont être découpées en sous-taches, générant ainsi une hiérarchie de taches. Chacune de ces taches va être exécutée selon un plan. Un plan est un ensemble de règles, compétences et connaissances. Ce plan va agir comme une contrainte sur les taches disant quelle sous-tache de quelle tache doit être exécutée. Chaque tache et ses sous-taches sont composées d'un plan. Cela va donc créer des relations temporelles (ou plus exactement, un ordonnancement temporel) entre la réalisation des taches.

Ensuite, chaque tâche est exprimée en termes d'objectifs. Ces objectifs ont des statuts et des conditions qui doivent être satisfaites pour dire qu'un objectif est atteint.

-GOMS (Goals, Operators, Methods, and Selection rules)

Ce modèle consiste à décrire les méthodes nécessaires à la réalisation d'objectifs/taches. Les méthodes sont vues comme une série d'étapes que l'utilisateur exécute. Cela permet de savoir comment une tache est réalisée. Ce modèle va donc être utilisé à l'exécution, lorsque l'utilisateur sera confronté à la réalisation d'une tache précédemment décrite.

Dans ce modèle, il se peut qu'un objectif puisse être réalisé par plusieurs méthodes. Si ce cas arrive, le système va faire appel à des règles de sélection pour définir quelle méthode est la plus appropriée.
Il y a deux niveaux de décomposition des taches. Le plus bas niveau de décomposition est la tâche que l'utilisateur veut consciemment effectuer. Le plus haut niveau de décomposition utilise des flux d'opérateurs qui vont contrôler l'exécution de la tâche.
Ce modèle fait la distinction entre une tache et une action. Une action est un opérateur qui est spécifié par des méthodes et associé avec des taches basiques. Un opérateur est une action cognitive et physique que l'utilisateur doit réaliser pour accomplir une tache.
Un avantage de ce modèle est que chaque opérateur est associé à un temps d'exécution permettant ainsi de prédire le temps nécessaire à la réalisation d'une tache.

-AMBOSS

Comme certains modèles présentés ci-dessus, celui-ci va découper les taches en arbre structuré. La différence ici est que cet arbre inclut des relations temporelles entre les taches et inclut aussi la description de ces taches.
Ce modèle de tâche est composé de taches, de "salles" (rooms), de rôles et de relations entre les taches.
Chaque tache a un type qui va permettre de savoir si elle est interactive entre l'utilisateur et le système, ou si c'est une action exécutée par le système comme vérifier un numéro de carte bleue.
Chaque tache a aussi des attributs "temps minimum" et "temps maximum" qui permettent de déterminer son temps d'exécution.
Chaque tache va avoir un indicateur de risque d'échec qui représente la probabilité que cet échec se produise et la probabilité que cet échec soit détecté et un facteur de risque.
Dans ce modèle, les taches doivent être exécutées séquentiellement. Ainsi toutes les sous-taches doivent être exécutées séquentiellement avant que la tâche qui régit ces sous-taches puisse être terminée.
Aussi, les taches peuvent communiquer entres elles en s'envoyant des messages. Ces messages peuvent être de simples messages d'information ou des messages qui vont permettre de déclencher des actions.
Les taches vont pouvoir appartenir à certaines salles. Le principe de ces salles repose plus ou moins sur le principe des packages en Java. Par exemple on peut interdire l'accès d'une classe à d'autres classes dans d'autres packages.
Un autre exemple serait les salles de jeux. On peut jouer à un type jeux en particulier dans une salle et à un autre type de jeux dans une autre salle mais on ne peut à aucun moment intervertir les jeux et les salles.

Pour en revenir au modèle, les taches peuvent êtres attachées à des salles ou elles peuvent être interdite à certaines salles. Ces salles ont un certain nombre de propriétés comme une description, un nombre maximum de personne et un flag qui indique si une salle est fermée ou ouverte.
Enfin, ce modèle permet de décrire différents acteur avec différents rôles. Ces rôles vont être associés à des taches particulières selon leurs caractéristiques.

Université Nice SOPHIA ANTIPOLIS

Polytech'Nice-Sophia
Département Sciences Informatiques

Adaptation des interfaces aux environnements

Comparaison technologique

PhoneGap vs Ionic

Tony BULTE

Le 7 novembre 2014

Introduction

Phonegap et Ionic sont tous les deux des Frameworks permettant de créer des applications cross-platform. C'est à dire qu'elles sont adaptable aux différents supports, elles vont pouvoir être utilisées sur n'importe quel smartphone ou encore sur n'importe quel ordinateur avec un unique projet. Ceci est très puissant et permet donc de toucher beaucoup de supports sans aucune modification sur le projet de base.
Ainsi, j'ai décidé de comparer ces deux Frameworks car j'ai déjà eu l'occasion de les utiliser plus ou moins tous les deux et qu'il permettent de faire des webapp très puissantes et utilisables sur différents supports. De plus, je trouve que ces deux Frameworks sont complémentaire. En effet, on peut utiliser Ionic en surcouche d'un projet Phonegap. Phonegap va permettre de définir la fonctionnalité de bases de l'application alors qu'Ionic va être plutôt orienté final UI.

PhoneGap vs Ionic

Apres cette bref description, essayons de confronter ces deux Frameworks sur le plan de la plasticité :

Critères \ Framework	PhoneGap	Ionic
Contexte d'usage	Peut etre utilisé sur tout type de support (mobile, tablette, PC)	Peut etre utilisé sur tout type de support mais fournis des jeux de look and feel desctiné à une utilisation sur des supports tactiles
Difficulté d'utilisation (Pour le développeur)	Phonegap est plus simple a prendre en main qu'Ionic. Il ne dépend d'aucune librairie, tout peut se faire en HTML/CSS/JS. Mais on peut aussi ajouter d'autres librairies comme JQuery mobile ou Sencha pour	Ionic est basé sur AngularJS. Ce n'est pas forcément aussi accessible que le simple javascript avec Phonegap. Pour 'styliser' l'application, il y a des classes CSS prédéfinies

	faire des applications plus poussées. Mais pour une utilisation basique, Phonegap n'est rien d'autre que du 'web développement'.	(Comparable à Bootstrap) qui sont très simple à utiliser.
Design et comportement	Ne fournit rien en termes d'UI. Tout ce qui touche aux interactions avec l'utilisateur doit être géré par le développeur, soit en mettant en place son propre CSS soit en utilisant des surcouches à Phonegap.	Met en place un système simple de mise en forme (comme du Bootstrap). Fournit des jeux de d'icônes, des jeux d'animation et de 'look and feel' qui ressemblent à du natif.
Méta-modèle (Comparaison avec Cameleon)	Pour le contexte d'usage, Phonegap se trouve au niveau de la Platform. En effet il permet d'accéder à des fonctionnalités "bas niveau" du téléphone comme l'accéléromètre ou encore l'appareil photo. Ensuite, au niveau du modèle UI, Phonegap se trouve à l'étage de l'abstract UI. En utilisant ce Framework, le développeur doit lui-même se charger de définir des modèles de taches, les fonctionnalités ou encore de définir l'interface graphique. L'avantage de phonegap est que l'on peut ajouter des surcouches telles que	Pour le contexte d'usage, Ionic est très orienté utilisateur final et se concentre beaucoup sur le look and feel. Au niveau du modèle UI, il se trouve entre la concrete UI et la final UI. En effet, Ionic permet de gérer des containers. On peut créer des groupes d'éléments d'interface graphique qui sont reliés à des fonctionnalité/taches communes. De plus, Ionic permet aussi de faire une interface graphique très avancée et détaillé, c'est pourquoi je pense que l'on peut aussi classer ce Framework dans l'étage final UI du modèle Cameleon.

	Ionic par exemple qui vont permettre de donner des effets interactifs qui paraissent natifs (menu latéral que l'on peut faire glisser,...)	Enfin, même si Ionic permet de faire un design très avancé, le développeur peut lui aussi ajouter son propre CSS à l'application et n'est obligé d'utiliser le design fournit. En revanche, pour tout ce qui concerne les modèles de taches, Ionic n'aide pas le développeur sur ce point.
Les "majeures" et les "mineurs"	Au niveau de l'UI, Phonegap ne fournit aucune API. J'ai déjà travaillé sur des projets (professionnels et personnels) utilisant Phonegap, et lorsque l'on utilise ce Framework on aime faire d'une pierre deux coup. C'est à dire que l'on va essayer de faire un design le plus responsive possible afin de pouvoir toucher les navigateurs web et les smartphones avec le minimum de modifications possibles entre les différents supports. En utilisant Phonegap, le développeur est censé se concentrer plus sur les fonctionnalités (le cœur de son application) comme	Ce Framework est très orienté UI interaction et animations natives. Il est possible d'utiliser ce Framework en surcouche d'un projet Phonegap Je pense que le fait de coupler les deux permet d'obtenir une application complète (Jolie et fonctionnelle). Contrairement à Phonegap, Ionic va vraiment être orienté smartphone. La majorité des éléments graphiques proposés sont des éléments que l'on retrouve beaucoup sur des applications smartphone et très peu sur des sites internet. Bien sûr on peut utiliser ce Framework en ayant pour cible les navigateurs web (PC) mais avec des interactions pas forcément adaptées à l'environnement d'utilisation. Par exemple les menus latéraux que l'on peut faire apparaitre en

	l'utilisation de l'appareil photo ou encore la géolocalisation plutôt que sur le design	faisant glisser son doigt de gauche à droite sur l'écran d'un smartphone ne seront pas très adaptés à une utilisation sur un navigateur web avec un souris.
Avantages	-Libertée totale pour le développeur -Facile à prendre en main (seule connaissance requise : JS) -Permet d'utiliser des fonctionnalités de base du téléphone (géolocalisation, appareil photo,...) -Le developpeur peut ajouter d'autres librairies pour enrichir son projet (graphiques ou fonctionnelles) -Plus connu qu'Ionic donc il y a beaucoup d'existant (doc, forum,...)	-Fournit beaucoup d'elements graphiques (icones, animations,...) -Le developpeur peut ajouter d'autres librairies pour enrichir son projet -Fournit un rendu très ergonomique et beau a l'utilisateur final
Inconvenients	-L'application finale peut contenir des bug graphiques (animations pas fluides, design incertain,...) -Les librairies que l'on peut ajouter a un projet Phonegap ne sont pas toutes aussi abouties les unes que les autres.	-Plus difficile à prendre en main car basé sur AngularJS -Si le developpeur utilise des elements graphiques fournit (ex : menu lateral caché) ceux-ci ne seront pas toujours adapté à des supports autres que tactiles -Documentation floue pour certaines fonctionalités, très peu

	-Il peut y avoir des problemes de rendu selon les platform (android, ios, blackberry,...)	connu sur les forums, difficile d'apprendre à l'utiliser en autodidacte

Pour conclure, en terme de développement, Phonegap va permettre d'avoir la main sur plus de fonctionnalités et va permettre d'avoir une application très modulable et personnalisable par le développeur. En revanche, Ionic est très bien pour déléguer le développement des interactions, des animations ou encore pour éviter la création d'icônes.

Pour conclure, ces deux Framework sont très complémentaires et n'ont donc pas grand-chose en commun. Les utiliser individuellement permet de faire une application cross-platform très simplement, mais utiliser ces deux Framework dans le même projet va permettre d'obtenir un résultat final très abouti avec quelque chose de fonctionnel et en même temps quelque chose de très adapté aux smartphone en terme de design, d'interaction et de comportement.