THESE

présentée en vue de l'obtention du

Doctorat de l'Université de Lille

Spécialité : **Informatique**

par

MICHEL DIRIX

Pour une Collaboration Efficace dans les Outils de Modélisation Logicielle

Soutenue le 4 Juillet 2016

JURY

Philippe Collet	Président
Jean-Marc Geib	Directeur de thèse
Xavier Le Pallec	Encadrant
Marianne Huchard	Rapporteur
Khalid Benali	Rapporteur
Alexis Muller	Examinateur

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille

Ecole Doctorale SPI
Université Lille Nord de France

Axellience
Innovation for software development

Je dédie ce mémoire à Raymonde et Michel qui auraient été si fiers.

Résumé

Depuis une vingtaine d'années, les logiciels ont atteint un stade très élevé de complexité. L'utilité de la modélisation logicielle s'est avérée de plus en plus importante, car elle s'abstrait de certaines préoccupations en ne fournissant que les informations relatives au point de vue souhaité, comme l'architecture ou les besoins utilisateurs. Toutefois l'activité de modélisation est une activité collaborative impliquant des développeurs, des architectes ou encore des clients. Avec la mondialisation, les équipes deviennent réparties à travers le monde impliquant des différences de culture, langage, fuseaux horaires.
Si les environnements de modélisation ont par le passé été souvent mono-utilisateurs, supporter logiciellement la collaboration est devenue depuis peu un réel sujet d'étude. C'est dans ce contexte que s'inscrit ma thèse. Elle a tout d'abord pour but de définir un système de communication constituant la base de toutes collaborations entre utilisateurs d'outils de modélisation. La thèse vise ensuite à rendre efficace cette collaboration en plaçant la conscience de celle-ci au sein des précédents outils à l'aide de fonctionnalités collaboratives identifiées ou conçues dans mes travaux. Ces derniers ont été appliqués à l'outil GenMyModel, un outil de modélisation en ligne enregistrant plus de 200 000 utilisateurs, pour valider les différentes études et conceptions réalisées.

Mots-clés :

Modélisation, Collaboration, Awareness, Expérience Utilisateur

.

Thèse préparée au sein du laboratoire CRIStAL
Université Lille 1
Bâtiment M3 extension
Avenue Carl Gauss
59655 Villeneuve d'Ascq Cedex FRANCE

Summary

Title : For an Efficient Collaboration in Software Modeling Tools

For twenty years, the softwares have reached a very high stage of complexity. The usefulness of the software modeling became increasingly important as models provide an abstraction of the software, and allows the software designer to describe information about important viewpoints such as the architecture or the user needs. With the globalization, teams are distributed over the world, implying culture, language and timezones differences. The modeling activity is collaborative since it implies many stackholders such as developers, architects or clients.
The modeling environments were historically single-user but the trend is to provide collaborative tools to address the aforementioned problems. The topic of my thesis is to provide an efficient collaboration. The first part deals with the definition of a communication system building the basis of all collaborations between the users in modeling tools. Then, I proposed a new and efficient collaboration by placing awareness at the center of modeling tools. This work has been implemented in the GenMyModel tool. GenMyModel is an online modeling tool registering more than 200.000 users which gave a perfect platform for the validation of the various studies and designs proposed in this thesis.

Keywords :

Modeling, Collaboration, Awareness, User eXperience

Remerciements

Après ces trois belles années, je tenais à remercier de nombreuses personnes.

Tout d'abord, je remercie **Marianne**, **Khalid** et **Philippe** d'avoir accepté de faire partie de mon jury ainsi que pour leur sympathie et disponibilité dans nos échanges.

Je remercie ensuite **Alexis** de m'avoir donné l'opportunité de faire cette thèse ainsi que sa confiance et **Vincent** pour nos nombreuses discussions qui m'ont donnée un tas d'idées.

Merci également à **Jean-Marc** qui m'a permis de prendre du recul et m'a grandement aidé lors de nos réunions. Tu m'as également guidé sur le bon chemin pour la rédaction de ce manuscrit.

Merci à **Bénédicte**, pour son aide et son efficacité !

Xavier, nous avons passé de nombreuses heures à faire avancer mes travaux, à refaire le monde et à trouver des solutions pour la suite de ma carrière. Merci pour le temps que tu m'as consacré et tous les moments partagés durant ces trois années. Je compte sur toi pour mener Carbon dans la bonne direction.

Jean-Claude, tu es la personne qui m'a le plus impressionnée par ta rigueur et l'ampleur de tes rêves. Je n'ai qu'une chose à te dire : réalises les !

Anne-So, garde ta bonne humeur qui fait plaisir à voir tous les matins !

Antoine, merci pour tous les moments de rigolade !

Elisa, un grand merci pour tout le temps que tu as passé pour moi, que ce soit sur la couverture de ce manuscrit, la vidéo de ma présentation...

Mickaël, en plus d'un collègue, j'ai trouvé un ami sur qui je peux compter. Nous avons partagé le même bureau pendant 2 ans et ce fut un réel plaisir. Nos chemins se séparent mais j'espère qu'ils se rejoindront très bientôt pour mener un projet commun.

Mes derniers mots s'adresseront à ma famille. Tout d'abord, mes **parents**, mes **beaux-parents** qui m'ont toujours soutenu et m'ont enseigné toutes leurs valeurs et leurs conseils que j'essaie d'appliquer au quotidien.

Enfin, un grand merci à **Géraldine**, **Cloé** et **Noémie**. Vous êtes toujours à mes côtés dans tout ce que j'entreprends. Ces trois années n'ont pas tous les jours été faciles mais vous avez toujours été présentes dans ces moments là. Et Géraldine, tu me soutiens à nouveau dans mon nouveau projet d'après-thèse, j'ai énormément de chance de t'avoir à mes côtés. J'espère pouvoir te rendre un jour tout ça. Je vous aime et merci encore <3

Table des matières

Table des figures

Liste des tableaux

Introduction

De nos jours, les logiciels deviennent de plus en plus volumineux et complexes. Il devient courant d'avoir des logiciels de plusieurs millions de lignes de code [DeFranco-Tommarello and Deek, 2002]. Cette complexité croissante apporte des contraintes liées à l'architecture du logiciel. Les logiciels deviennent plus difficiles à concevoir et à maintenir. Pour adresser cette complexité, la modélisation logicielle (pour la suite de la thèse, nous parlerons simplement de modélisation) prend de l'importance. Le but d'un modèle est de fournir une abstraction d'un système, permettant aux ingénieurs de raisonner sur ce système en ignorant les détails et en se concentrant sur les informations les plus importantes [Brown, 2004]. A titre d'exemple, une carte routière est un modèle. Il s'agit d'une représentation abstraite d'un système (un territoire géographique) qui a pour but de faciliter les trajets en donnant des informations sur les routes telles que leur type. Les informations annexes au territoire géographique comme le nom des habitants, la langue parlée, etc, ne sont pas représentées.

Plusieurs langages de modélisation existent et sont spécifiques à une préoccupation particulière (Interaction, logique, déploiement, métier). Dans le cadre de cette thèse, nous nous concentrerons donc sur le langage UML (Unified Modeling Language).

La conception d'un logiciel est de surcroit une activité collaborative impliquant de nombreux acteurs comme les développeurs, les architectes, les clients, etc [Jiménez et al., 2009]. Et cette activité devient maintenant distribuée. En effet, depuis une dizaine d'année, dans le but de réduire les coûts de développement d'un logiciel ou de cibler de nouveaux marchés, les entreprises ont tendance à sous-traiter [Hossain et al., 2011]. Cette tendance a été appelée Global Software Development (GSD). Mais qu'est-ce que collaborer ? Avant de donner les définitions exactes dans la seconde partie de cette thèse, une description rapide de la collaboration est l'activité de travailler ensemble dans le but d'accomplir une tâche commune. Collaborer se fait généralement dans un rayon proche. La pratique du GSD n'est pas sans problème. Les équipes étant réparties à travers le monde, les fuseaux horaires, les cultures, les langues, peuvent être différentes et poser des problèmes de communication et de synchronisation.

GenMyModel n'a pas échappé à ces problématiques. GenMyModel est un outil UML en ligne [Axellience, 2015][Dirix et al.,], lancé en version béta en Septembre 2012. L'outil a pour vocation à régler les problèmes

d'installation, de localisation des modèles et de compatibilité de versions en se déployant dans le Cloud. Peu après son lancement, les utilisateurs ont fait ressentir à Axellience (société éditrice de GenMyModel) le besoin de collaborer en même temps sur les modèles. A partir de cette demande, j'ai rejoint la société Axellience pour travailler sur cette problématique. GenMyModel compte aujourd'hui environ 200 000 utilisateurs répartis dans plus de 200 pays. Il était donc primordial de fournir à nos utilisateurs une collaboration adaptée à leurs besoins. C'est pourquoi pour cette thèse, j'ai décidé de placer l'utilisateur au cœur de mon travail. Celui-ci a été impliqué dans toutes les phases de manière à effectuer une recherche en adéquation avec les problématiques de la réalité.
GenMyModel n'est pas le seul outil à proposer la collaboration à ses utilisateurs. Comme je le montrerai plus tard dans cette thèse, de grands acteurs apportent également cette nouvelle fonctionnalité comme IBM ou encore Google. Cette thèse se déroule dans un contexte industriel et amène donc des problématiques de performances, de passage à l'échelle, de maintenance et de généricité. En effet, à l'origine du projet, l'éditeur en place n'avait qu'une seule vocation : proposer un éditeur UML. Néanmoins, il était déjà prévu d'ajouter d'autres éditeurs pour la gestion des base de données ou encore BPMN. Cette contrainte imposait donc de mettre en place des solutions génériques notamment pour le processus de mise en place de la collaboration entre utilisateurs. Le système doit s'avérer être le même que ce soit pour UML ou un autre langage. De plus, la solution doit être facilement extensible et maintenable. L'ajout de nouvelles fonctionnalités ne doit pas remettre en cause tout le système. Lors d'une édition collaborative, des conflits peuvent survenir lors de changements concurrents, il est primordial que cette gestion des conflits soit simple pour les utilisateurs pour éviter une perte de temps.
Apporter la collaboration aux utilisateurs ne signifie pas seulement les autoriser à collaborer. Le fait de travailler de seul à plusieurs implique une prise de conscience de l'utilisateur. Il n'est plus seul, il doit prendre en compte les actions des autres sur un environnement qui devient partagé. Cette prise de conscience est appelée *awareness*[Herrmann et al., 2013]. Cette dernière permet une collaboration efficace
[Lauwers and Lantz, 1990]. Par efficace, nous entendons le fait que chaque utilisateur ait une connaissance des interactions les plus importantes avec l'environnement partagé dans le but de mener à bien le projet commun. Cette thèse a pour objectif de montrer comment rendre cette collaboration efficace au travers de fonctionnalités collaboratives de manière à ce que les utilisateurs puissent accéder à cette connaissance commune. Avant de rendre la collaboration efficace, un travail préalable est néces-

saire : permettre la collaboration entre les utilisateurs. Pour cela, l'étude des différentes phases composant la collaboration a été nécessaire avant de pouvoir proposer une réponse adaptée pour tous les outils de modélisation.

Les contributions apportées sont les suivantes :

C1 : Contributions sur la mise en place de la collaboration

1. Un système de commandes pour permettre l'échange de modifications entre utilisateurs : dAMOCleS.
2. Passage à l'échelle du système dans GenMyModel.

C2 : Contributions sur l'efficacité de la collaboration

1. Définition des différents modes de collaboration.
2. Classement des informations d'awareness à afficher pour chaque mode de collaboration.
3. Définition de différents contextes impactant les besoins.
4. Implémentation de l'historique des modifications.
5. Etude sur les usages et points d'amélioration de l'historique.

Cette thèse est de ce fait au croisement de trois domaines de recherche :

— Model-Driven Development(MDD) : mes travaux s'appliquent dans le cadre de GenMyModel, un outil de modélisation.

— Computer-Supported Cooperative Works (CSCW) : domaine principal pour tous les travaux autour de la collaboration.

— Human-Computer Interactions (HCI) : en plaçant l'utilisateur au centre des travaux, il faut pouvoir évaluer ses impressions et son expérience vis à vis des fonctionnalités ajoutées.

Pour détailler la mise en place d'une collaboration efficace pour les outils de modélisation, le document se structure en deux parties : une pour chaque contribution (C1 et C2).

Première partie

Permettre la communication des changements entre les utilisateurs

1 Etude des différentes phases

Lors de ce chapitre, je commencerai par expliquer le fonctionnement des deux types de collaboration : synchrone et asynchrone. A partir de cette explication, nous pourrons définir les différentes phases nécessaires pour la mise en place de la collaboration. Enfin, nous étudierons les différents travaux/outils traitant de ces différentes phases.

1.1 Modélisation synchrone ou temps-réel

La collaboration synchrone a souvent comme synonyme collaboration temps-réel. Par temps-réel, je n'entends pas le fait d'être le plus rapide possible dans le sens de l'instantanéité des actions mais plutôt le fait que chaque action d'un utilisateur est répercutée chez les autres utilisateurs connectés au projet. De nos jours, les personnes sont habituées à recevoir automatiquement des informations. Nous utilisons tous les jours ce genre d'application à l'instar de Twitter ou de Facebook. En utilisant ces applications, nous n'avons plus à actualiser les pages web ou les applications pour obtenir les dernières actualités[Rai, 2013]. L'essor de ce type d'application vient de l'émergence de nouvelles technologies du web à l'instar de NodeJS et des nombreux frameworks JavaScript. Avec ces langages, il est désormais plus facile d'échanger des informations entre le client et le serveur. Dans le cas de la modélisation collaborative synchrone, les utilisateurs récupèrent le modèle sur un référentiel de modèles. Ensuite, un système de communication est créé permettant l'échange de données entre les utilisateurs et le serveur. Dès lors qu'un changement est réalisé, il est tout de suite envoyé au serveur et celui-ci se charge de renvoyer le changement à tous les clients connectés sur le projet. La figure 1.1 montre ce fonctionnement en prenant comme référentiel de modèle un serveur distant.

La modélisation collaborative en temps-réel a été décrite comme une solution future [young Bang et al., 2012] en 2012, elle reste donc nouvelle et peu exploitée dans les outils actuels.

1.2 Modélisation asynchrone

Lorsque des utilisateurs travaillent de manière asynchrone, ils vont d'abord commencer par demander au serveur (ou un gestionnaire de sources) à récupérer le modèle. Ensuite, ces utilisateurs vont chacun effectuer des modifications de leur côté. Puis, l'un des clients va envoyer

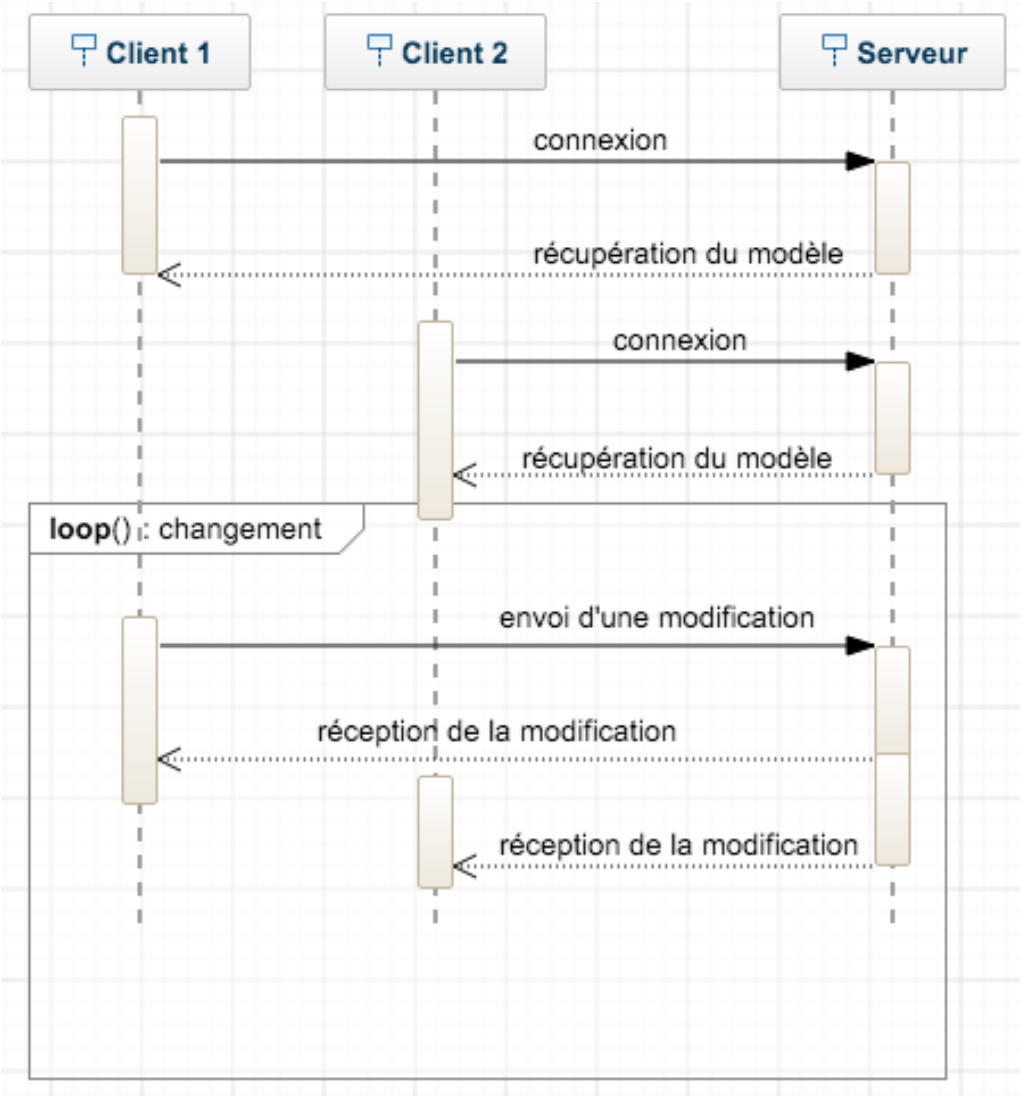


FIGURE 1.1 – Fonctionnement général de la collaboration synchrone

ses modifications au serveur. Lorsque la seconde personne souhaite envoyer ses modifications au serveur, celui-ci le notifie que des modifications ont eu lieu par rapport à la version qu'elle possédait avant (modifié par le premier utilisateur) et elle doit donc les récupérer. Cette récupération des modifications engendre une fusion (appelé couramment *merge*) du modèle local de l'utilisateur avec le modèle distant, présent sur le serveur. Ce merge peut aboutir éventuellement à des conflits dans le cas où les utilisateurs auraient travaillé sur les mêmes éléments de modèle. Il est alors demandé à l'utilisateur de gérer ces conflits avant d'envoyer les

modifications fusionnées au serveur. Ce processus est représenté sur la figure 1.2.

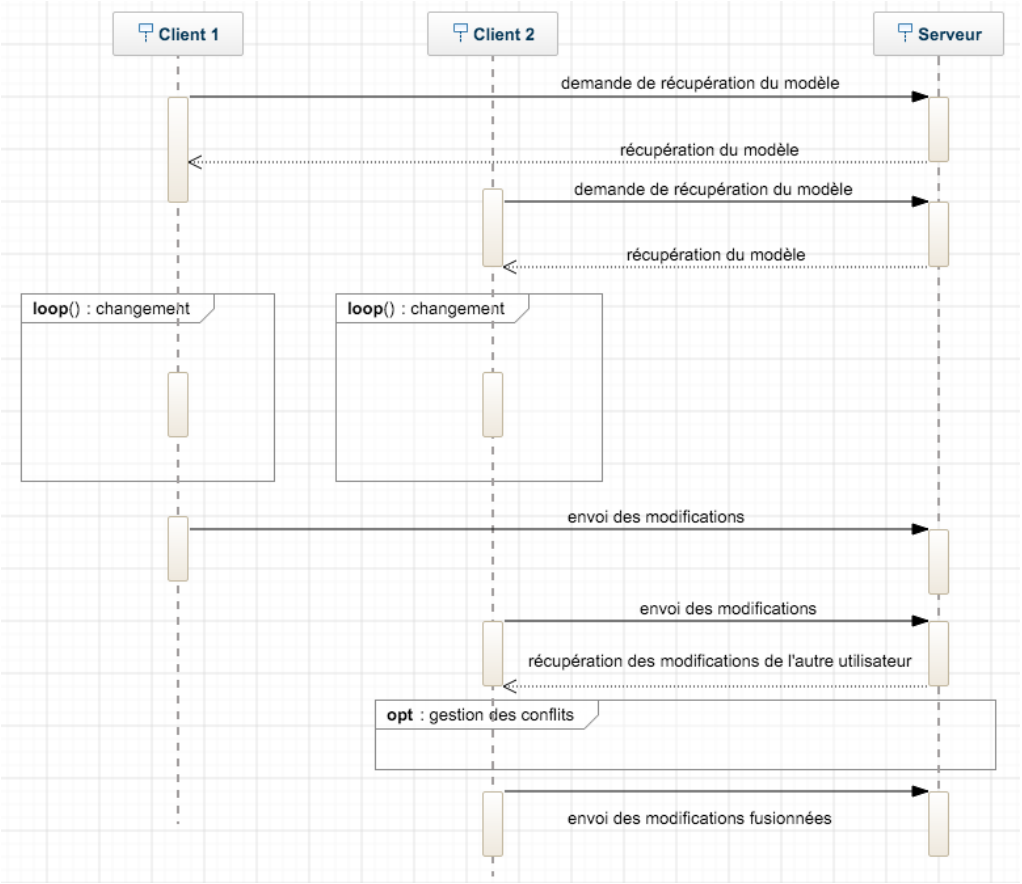


FIGURE 1.2 – Fonctionnement général de la collaboration asynchrone

Ce type de collaboration est déjà largement utilisé dans le domaine du développement logiciel. En effet, la plupart des équipes travaillant sur des projets, collaborent sur le code de cette manière en utilisant des dépôts de code à l'instar de Git ou de SVN.

1.3 ETAPES À ANALYSER

En confrontant les deux types de collaboration, comme le montre la figure 1.3, nous pouvons déterminer les grandes étapes à prendre en compte dans le but de gérer tout le processus de la collaboration.

Que ce soit lors d'une collaboration synchrone ou asynchrone, les utilisateurs doivent récupérer à un moment donné le modèle. Cette récupéra-

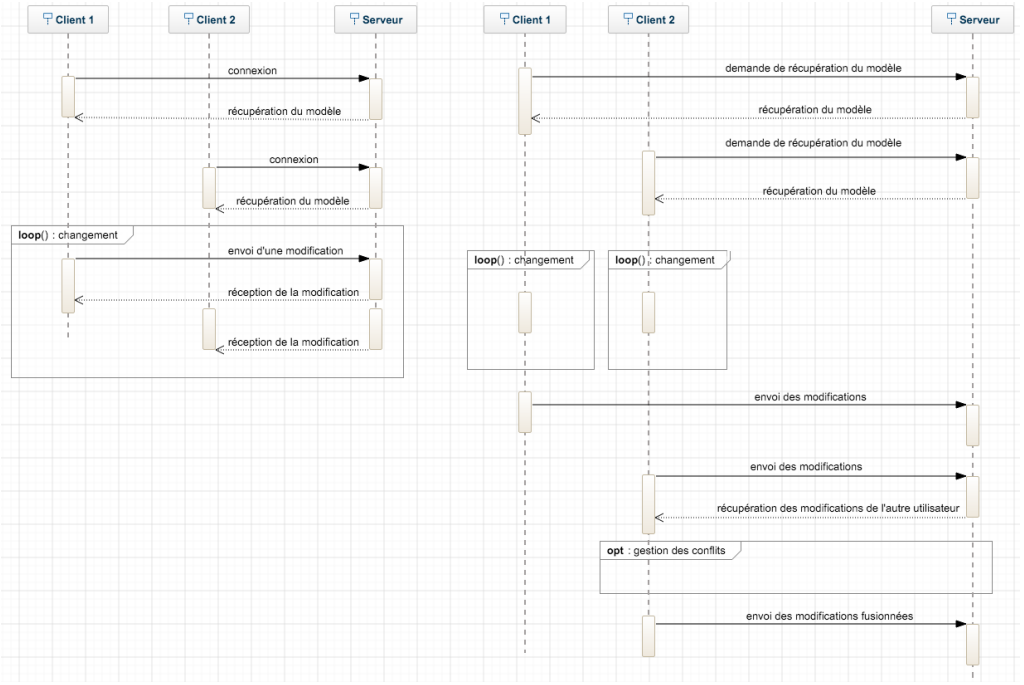
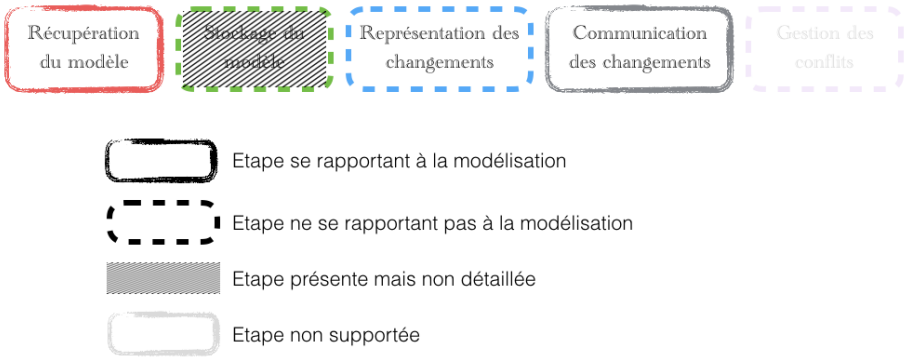


FIGURE 1.3 – Différences entre le fonctionnement synchrone et l'asynchrone

tion du modèle implique également de savoir comment stocker celui-ci. De plus, même si la fréquence d'envoi n'est pas la même, les utilisateurs doivent envoyer leurs modifications au serveur. Il est donc primordial d'avoir une représentation de ces modifications, mais aussi de déterminer par quel moyen celles-ci sont échangées entre les clients et le serveur. Enfin, pour la collaboration asynchrone, une notion de conflits apparait et ceux-ci doivent être gérés par les utilisateurs. Lors de la collaboration synchrone, ce ne sont pas les utilisateurs qui vont gérer les conflits possibles mais le système lui-même, puisqu'il doit garantir l'ordonnancement des messages échangés. Pour résumer, la solution à adopter pour fournir tout le processus collaboratif doit satisfaire les points suivants :

A partir de ces différentes étapes, plusieurs cas sont à envisager. Il se peut que l'étape soit bien supportée mais je n'ai pas pu en déterminer le fonctionnement. Dans ce cas, par rapport au schéma présenté précédemment, l'étape apparaitra hachurée. Lorsque la solution présentée ne se rapporte pas à un outil de modélisation, les bordures des diffé-

rentes étapes seront en pointillées, pleines en rapport avec la modélisation. Dans le cas où l'étape n'est pas supportée, elle apparaitra grisée. Le schéma suivant synthétise la légende adoptée.



Ce schéma sera utilisé tout au long de l'état de l'art à venir pour identifier le support des différentes phases par les travaux et outils présentés.

1.4 Exemple conducteur pour la représentation des modifications

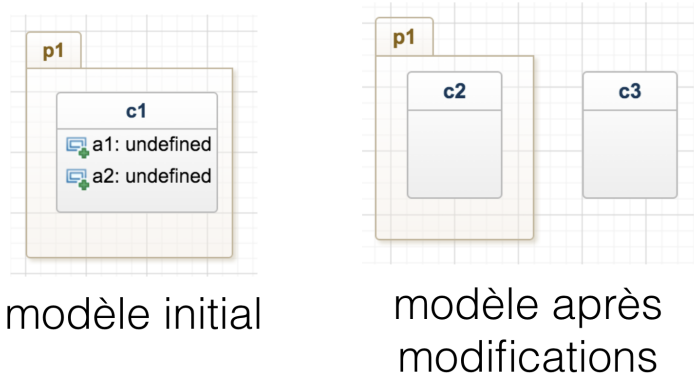


Figure 1.4 – Exemple conducteur pour la représentation des modifications

La figure 1.4 montre un modèle initial à gauche et le modèle que nous souhaitons après modifications. Le modèle initial contient un package *p1* ayant pour identifiant *package1_id*, contenant une classe *c1* avec *class1_id* pour identifiant. Cette classe contient deux attributs *a1* et *a2*. Sur ce modèle, nous lui appliquerons les modifications suivantes :

— Suppression de la classe *c1*

— Création de deux nouvelles classes *c2* et *c3*

— Ajout de la classe *c2* dans le package *p1*

Dès lors qu'une solution présentée traitera de la représentation des modifications, je présenterai comment sont structurées les modifications pour s'appliquer à l'exemple présenté ici.

1.5 Support des différentes phases par les outils et travaux de recherche

Dans cette section, nous allons étudier différentes solutions présentes dans la littérature mais aussi des outils commerciaux, utilisés par un grand nombre d'utilisateur, afin de déterminer comment sont supportées les différentes phases de la collaboration. Ne seront présentées que les informations présentes dans les articles ou ayant pu être déterminés en testant les outils commerciaux. Les solutions ne couvrent généralement pas toutes les phases de la collaboration. Dès lors que leur support n'est pas mentionné, cela signifie que les informations ne sont pas disponibles.

D-Praxis

Phases supportées :

Une première approche, orientée pair à pair consiste en une réplication du modèle en local pour chaque développeur, la synchronisation se faisant au travers de messages [Saito and Shapiro, 2005]. D-Praxis utilise cette approche [Mougenot et al., 2009].

Description

Lors de l'utilisation de D-Praxis, chaque utilisateur travaille en local sur une partie du modèle uniquement et décide à un moment d'envoyer ses modifications aux autres personnes du groupe. Néanmoins chaque action peut impacter le travail des autres si plusieurs utilisateurs partagent des éléments de modèles communs entre leurs différents modèles locaux. Les conflits pouvant survenir sont les suivants :

— Valeur de propriétés différentes : deux clients changent la valeur d'une propriété d'un des éléments de modèle par une valeur différente.

— Ajout et suppression de la valeur d'une propriété : un client ajoute une valeur sur une propriété et un autre client la supprime.

— Valeur de références différentes : deux clients changent la valeur d'une référence d'un des éléments de modèle par une valeur différente.

— Ajout et suppression de la valeur d'une référence : un client ajoute une valeur sur une référence et un autre client la supprime.

— Suppression d'un élément de modèle et ajout d'une valeur sur une référence ciblant l'élément de modèle : un client supprime un élément de modèle et un autre client le cible à partir de la valeur d'une référence.

— Suppression d'un élément de modèle et ajout d'une valeur sur une référence à partir de l'élément de modèle : un client supprime un élément de modèle et un autre client ajoute une référence à partir de cet élément de modèle.

— Suppression d'un élément de modèle et ajout d'une valeur sur une propriété à partir de l'élément de modèle : un client supprime un élément de modèle et un autre client ajoute une propriété à partir de cet élément de modèle.

Ces conflits semblent logiques mais n'avaient pas été répertoriés. Pourtant, il s'agit de conflits pouvant survenir pour n'importe quel outil de modélisation. Après cette identification, les auteurs de D-Praxis ont ensuite défini 6 opérations permettant de modifier le modèle à partir des travaux de Mens [Mens, 2002] :

— create(me,mc) créé une instance de l'élément de modèle *me* de la méta-classe *mc*. Un élément de modèle peut être créé si et seulement s'il n'existe pas dans le modèle (des UUID sont utilisés pour empêcher d'avoir deux éléments de modèles avec le même identifiant) ;

— delete(me) supprime l'élément de modèle *me*. Un élément de modèle peut être supprimé si et seulement s'il existe dans le modèle et s'il n'est pas référencé par un autre élément de modèle. Quand un élément est supprimé, toutes ses propriétés sont automatiquement supprimées ;

— addProperty(me,p,v) attribue une valeur v à une propriété p de l'élément de modèle *me* ;

— remProperty(me,p) supprime la valeur, si elle existe, de la propriété p de l'élément de modèle *me* ;

— addReference(me,r,met) attribue un élément de modèle ciblé à une référence *r* à l'élément de modèle *me*.

— remReference(me,r) supprime l'élément de modèle ciblé, s'il existe, de la référence *r* de l'élément de modèle *me*.

Représentation des changements

Les opérations décrites précédemment constituent **le format de la représentation des changements**. L'atout majeur de cette représentation réside dans la généricité de la solution. En effet, celle-ci ne dépend pas du type des éléments modifiés, ni du type de la propriété. Les opérations à appliquer pour notre exemple, figure 1.4 sont les suivantes :

```
delete(class1_id)
create(class2_id, Class)
create(class3_id, Class)
addProperty(package1_id, "packageableElement",
   class2_id)
```

Communication des changements

Lorsqu'un utilisateur souhaite partager ses modifications au groupe, celles-ci ne sont pas envoyées à tous les autres utilisateurs mais uniquement à ceux avec qui il partage des éléments de modèles communs. Ce qui signifie que si un utilisateur modifie une classe et décide d'envoyer ses modifications, tous les autres utilisateurs ayant cette classe dans leur modèle recevront les modifications.

Gestion des conflits

La gestion des conflits s'effectue à la réception des changements en se basant sur deux principes : une horloge de Lamport permettant d'ordonnancer les changements ayant eu lieu [Lamport, 1978] ainsi que la sémantique de suppression de D-Praxis [Blanc et al., 2008]. Lorsque deux opérations sont en conflit, la plus récente est conservée à moins que ce ne soit une suppression. Le problème majeur émanant de cette solution est la perte de changements. L'utilisateur n'est pas notifié des conflits pouvant survenir. Par exemple, si une première personne travaille longuement sur une classe en y ajoutant toutes les informations nécessaires, et qu'une seconde la supprime par mégarde, la première a donc perdu tout le travail qu'elle venait d'effectuer.

Résumé

Le principal atout de D-Praxis repose sur la représentation des changements qui est générique. Ceci permet de développer plus facilement l'éditeur pouvant faire communiquer les différents collaborateurs, mais aussi de faciliter la maintenance.

COMA : ajout d'un négociateur

Phases supportées :

Description

COMA [Rittgen, 2008] ajoute une nouvelle manière de gérer l'acceptation d'une nouvelle version du modèle par l'un des collaborateurs au travers d'un système de vote de manière à ce qu'**il gère les conflits**. Deux systèmes sont possibles : un vote à la majorité ou la définition d'un facilitateur qui prendra seul la décision d'accepter ou non la totalité des changements effectués. Lorsqu'un collaborateur soumet une nouvelle version, deux cas sont donc possibles, sa version est refusée et des commentaires sont ajoutés pour justifier le refus, ou sa version est acceptée. L'ajout d'un négociateur peut s'avérer problématique dans certains cas en fonction de l'intrusivité de la fonctionnalité et du rythme de travail de l'équipe. Dans le cas où un unique négociateur est présent, si le rythme de travail est trop élevé, il risque de ne pas pouvoir mettre à jour le modèle principal assez rapidement et ainsi ralentir toute l'équipe. De plus, si le système est trop intrusif pour le négociateur ou s'il est trop souvent sollicité, il sera sans cesse coupé dans son travail. A l'inverse, si le système ne lui notifie pas l'arrivée de nouvelles décisions à prendre, l'ensemble de l'équipe sera impacté puisque le négociateur ne pourra pas réagir rapidement.

Gestion des conflits

L'ajout d'un négociateur a été réalisé dans le but de gérer les conflits. Au lieu d'avoir cette gestion pour tous les collaborateurs, elle peut être réduite à une personne en particulier. Le manque de recul sur l'implémentation et le manque de détail sur l'intrusivité ne font pas de COMA une solution utilisable pour une collaboration durant laquelle un rythme de travail modéré ou important a lieu.

Delta Operation Language

Phases supportées :

Description

Delta Operation Language(DOL) [Kuryazov and Winter, 2014] permet de modifier directement le modèle à partir d'un langage. Le premier but de ce langage est de montrer les différences entre deux versions d'un modèle par des opérations décrites avec le langage, sur un diagramme d'activité uniquement. Les auteurs prévoient de l'utiliser pour supporter la collaboration sans préciser la nature synchrone ou asynchrone. Le langage servira donc de **représentation des modifications**.

Représentation des modifications

Un exemple d'utilisation est la suivant :

```
g1 = createInitialNode();
g3 = createOpaqueAction("Receive Order");
g2 = createControlFlow(g1,g3);
g5 = createActivityFinalNode();
g5.delete();
```

Sur cet exemple, un *InitialNode* a été créé, puis une *OpaqueAction*, pour ensuite créer un *ControlFlow* entre l'*InitialNode* et l'*OpaqueAction*. Enfin, un *ActivityFinalNode* a été ajouté, avant d'être supprimé.

Un éditeur souhaitant utiliser le langage pour manipuler les éléments de modèle se confronte à un problème : la généricité. La création d'un *ForkNode* ou *JoinNode* par exemple sont ici deux actions bien distinctes contrairement à la solution de D-Praxis. Ce manque de généricité force l'éditeur à traiter les actions cas par cas, complexifiant le développement de l'éditeur et sa maintenance. Le nombre d'actions possibles n'est pas très volumineux. Par contre, si une telle solution est mise en place sur le diagramme de classe, il devient très coûteux en temps de développer et de maintenir l'éditeur se servant de ce langage. De plus, le langage ne permet que de manipuler le modèle et non la partie graphique. Si la langage doit être étendu pour la prendre en compte, le nombre d'actions possibles sera encore plus élevé. Si l'on se projette sur cette solution ap-

pliquée au diagramme de classe et à notre exemple illustré par la figure 1.4, les modifications seraient représentées de la manière suivante :

```
c1.delete();
c2 = createClass("c2");
c3 = createClass("c3");
p1.addClass(c2);
```

Résumé

DOL souffre du manque de généricité et complique sa mise en place dans un outil de grande envergure. Néanmoins, son principal avantage est d'être facilement lisible par un humain.

Obeo designer : une nouvelle initiative

Phases supportées :

Description

En décembre 2015, une présentation a été réalisée à la SiriusCon montrant le désir d'Obeo de rendre leur outil *Obeo Designer* collaboratif [1]. Dans cette présentation, l'auteur montre que Obeo a pour objectif d'être collaboratif de manière synchrone par un système de *lock*. Néanmoins, le détail de la gestion de la collaboration n'est pas détaillé.

Gestion des conflits

Le système de *lock* permet d'éviter aux utilisateurs la gestion de conflits en leur interdisant l'accès concurrent aux mêmes éléments de modèle.

Résumé

Cette initiative manque de détails. Néanmoins, elle montre l'intérêt récent porté à la modélisation collaborative et conforte par ailleurs les travaux de thèse présentés ici.

1. http ://fr.slideshare.net/pcdavid_/collaborative-modeling-with-sirius

Les spécifications XMI

Phases supportées :

Description

La partie *Transmitting Incomplete Metadata* des spécifications XMI mentionnent l'envoi de modifications [OMG, 2015] au travers de fragments XMI. Ces fragments correspondent à l'élément modifié lors d'une suppression, d'un changement de propriété ou d'un changement de position (modification sans changement de type et de conteneur)

Représentation des changements

Les fragments définis dans la spécification permettent d'adresser la représentation des changements. Je vais détailler ici le format des données à envoyer par rapport à l'exemple défini sur la figure 1.4.

Au départ de notre exemple, le fichier XMI correspondant au modèle est le suivant :

```
<xmi:XMI xmlns:uml="http://www.omg.org/spec/UML/20110701"
xmlns:xmi="http://www.omg.org/spec/UML/20110701" xmi:id="
   model_id">
  <uml:Package xmi:id="package1_id" xmi:label="p1">
    <packagedElement xmi:type="uml:Class" xmi:id="
       class1_id" name="c1">
      <ownedAttribute xmi:type="uml:Property" name="a1"/>
      <ownedAttribute xmi:type="uml:Property" name="a2"/>
    </packagedElement>
  </uml:Package>
</xmi:XMI>
```

Les modifications à apporter à ce modèle se présentent de la manière suivante :

```
<xmi:XMI xmlns:uml="http://www.omg.org/spec/UML/20110701"
 xmlns:xmi=" http://www.omg.org/spec/UML/20110701" xmi:id=
    "model_id">
  <difference xmi:type="xmi:Delete">
    <target href="original.xml#class1_id"/>
  </difference/>
  <difference xmi:type="xmi:Add" addition="Class_2 Class_3
     ">
    <target href="original.xml#model_id"/>
  </difference>
  <packagedElement xmi:type="uml:Class" xmi:id="class2_id"
      name="c2">
  <packagedElement xmi:type="uml:Class" xmi:id="class3_id"
      name="c3">
  <difference xmi:type="xmi:Add" addition="Class_2">
    <target href="original.xml#package1_id"/>
  </difference>
</xmi:XMI>
```

Les modifications sont envoyées en même temps et l'élément XML *target* cible l'élément de modèle impliqué par la modification. Il n'est pas possible de créer d'élément de modèle directement. C'est pour cette raison que les éléments *Class_1* et *Class_2* sont présents dans le fragment. L'utilisateur à l'origine des actions doit d'abord réaliser la création de l'élément avant de l'ajouter dans le fragment envoyé. Après application des modifications, le modèle devient :

```
<xmi:XMI xmlns:uml="http://www.omg.org/spec/UML/20110701"
 xmlns:xmi="http://www.omg.org/spec/UML/20110701" xmi:id="
    model_id">
  <uml:Package xmi:id="package1_id" xmi:label="p1">
    <packagedElement xmi:type="uml:Class" xmi:id="
       class2_id" name="c2">
  </uml:Package>
  <uml:Class xmi:type="uml:Class" xmi:id="class3_id" name=
     "c3">
</xmi:XMI>
```

Cette solution impose donc que les modifications et notamment les créations se fassent côté client. Ceci peut engendrer des conflits pour les utilisateurs, difficiles à gérer. Prenons l'exemple où deux personnes travaillent ensemble sur une classe. La première personne supprime cette classe. Sa modification est réalisée. Une seconde personne dans le même

temps modifie le nom de cette classe. Sa modification va également être appliquée en local. Néanmoins, cette classe n'existe plus.

Résumé

Les spécifications XMI imposent que les actions soient d'abord réalisées côté client. Ce mode de fonctionnement apporte des conflits, non traitées. La granularité d'envoi des informations ne semble pas non plus adaptée. Pour éviter aux utilisateurs la gestion de conflits, il serait plus pertinent d'envoyer les modifications de manière atomique.

SLIM

Phases supportées :

Description

SLIM[Thum et al., 2009] est un outil web pour l'édition de modèles collaborative de manière synchrone. Thum et al. présentent l'architecture de SLIM adoptée pour la mise en place de cette collaboration.

Récupération du modèle

Lorsqu'un utilisateur rejoint le projet, un utilisateur déjà connecté est sélectionné au hasard pour lui envoyer la version actuelle du modèle. Le modèle lui est sérialisé à partir du XMI le représentant. Le modèle est stocké uniquement côté client dans le but de permettre aux utilisateurs de travailler sans être connectés à internet.

Stockage du modèle

Les travaux sur SLIM ne présentent pas de quelle manière est stocké le modèle. Il est dit que le serveur ne sert que de relai entre les clients. De ce fait, cela nous permet de supposer que les modèles sont stockés chez les différents clients. Dans ce cas, que se passe t-il si le premier client à se connecter n'a pas participé à la dernière session de collaboration.

Représentation des modifications

SLIM ne précise pas exactement sous quelle forme sont envoyées les modifications. Néanmoins, les auteurs parlent d'envoi incrémental des changements pour limiter le trafic du réseau.

Communication des modifications

Pour transmettre les modifications entre les différents clients, SLIM utilise le long-polling[Loreto et al., 2011]. Cette technique permet de conserver une connexion HTTP active et le client n'a pas à appeler très fréquemment le serveur pour lui transmettre des messages. La mise en place du long-polling est effectuée grâce à CometD [2] et en particulier *Jetty Comet Daemon*.

Gestion des conflits

De la même manière qu'Obeo, SLIM adopte une approche par un système de *lock*.

Résumé

La solution proposée ici apporte quelques limitations à la collaboration. En stockant le modèle chez les clients, la récupération du modèle n'est pas possible pour un utilisateur n'ayant pas participé à la session de collaboration précédente.

Le détail de la représentation des modifications n'est pas disponible même s'il est dit que seules les opérations atomiques sont envoyées à chaque modification.

GEMSjax

Phases supportées :

Description

GEMSjax[Farwick et al., 2010] est présenté comme un outil en ligne pour définir des méta-modèles. Les auteurs présentent brièvement com-

2. https ://cometd.org/

ment est implémenté l'éditeur ainsi que le fonctionnement de la collaboration.

Récupération du modèle

Le récupération du modèle s'effectue de la même manière que la communication des changements. Les auteurs ont mis en place dans leur prototype Comet, tout comme SLIM. De cette façon, lorsqu'un utilisateur se connecte, un événement est envoyé sur le bus CometD de manière à ce que le serveur lui envoie le contenu du modèle par ce même bus dans un format JSON (JavaScript Object Notation).

Stockage du modèle

Les modèles sont stockés dans un format EMF [3] (Eclipse Modeling Framework) sur le disque. Ce qui nécessite une conversion du modèle vers un format JSON pour pouvoir être envoyé à l'utilisateur.

Représentation des modifications

La représentation des modifications n'est pas détaillée. Néanmoins, nous pouvons supposer que le modèle client est sérialisé pour être envoyé au serveur puisque dans l'un des exemples, les auteurs annoncent que le client reçoit une version mise à jour après une modification effectuée par un autre utilisateur.

Communication des changements

De la même manière que pour la récupération du modèle, les échanges concernant les changements passent par un bus Comet.

Gestion des conflits

La gestion des conflits est très peu expliquée. En effet, les auteurs expliquent que pour éviter la duplication des modifications, ils ont mis en place des règles de priorité, sans les énoncer. Ils mentionnent également que lorsqu'un utilisateur sélectionne un élément de modèle, une notification est envoyée aux autres utilisateurs pour éviter les conflits. Nous ne savons pas si un système de *lock* est utilisé à partir de cette information ou si l'utilisateur est juste averti visuellement des éléments de modèle en cours de modification.

3. https ://eclipse.org/modeling/emf/

Résumé

GEMSjax rentre peu dans les détails sur l'ensemble de la mise en place de la collaboration. Néanmoins, l'utilisation de CometD semble être une solution pour les différents échanges de message et pas uniquement pour l'échange des modifications.

LucidChart

Phases supportées :

Description

LucidChart n'est pas réellement un outil de modélisation mais plutôt un outil de dessin proposant la collaboration synchrone. En effet, avec LucidChart, il est possible de réaliser un schéma n'ayant aucun sens UML. Par exemple, la figure 1.5 montre un schéma possible.

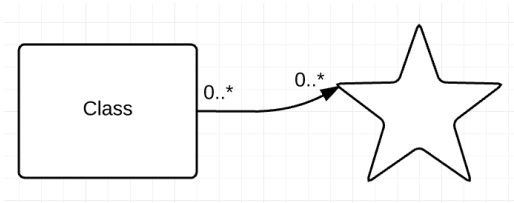


FIGURE 1.5 – Exemple d'un schéma non valide au sens UML avec LucidChart

Même si Lucid Chart n'est pas un outil de modélisation en tant que tel, graphiquement les concepts sont proches.

Représentation des changements

En observant les échanges réseau dans le navigateur au moyen des outils de développement, il est possible d'analyser le contenu des messages échangés pour en déterminer *la représentation*. Ceux-ci sont en format JSON. Pour la création d'un élément, le message envoyé est de la forme suivante :

```
[{"Action":"CreateBlock","Class":"TypeElement","Properties
   ":{"BoundingBox":{"x":xPosition,"y":yPosition,"w":
   largeur,"h":hauteur}, ... ,"Page":"
   IdentifiantDeLaFeuille","id":"IdentifiantDeLElement"}]
```

Pour une suppression, le message est décrit de la manière suivante :

```
"[{"Action":"RemoveBlock","id":"IdentifiantDeLElement","
   Properties":{...},"Class":"TypeElement","Page":"
   IdentifiantDeLaFeuille","Comments":[]}]"
```

Ces messages contiennent le type d'action à effectuer, ainsi que le type d'élément à créer. Différentes propriétés sont aussi ajoutées comme le positionnement de l'élément ou ses dimensions. De plus, l'identifiant du diagramme et l'identifiant de l'élément sont contenus dans le message. L'inconvénient du format de messages utilisé pour la création par LucidChart réside dans le grand nombre de propriétés passées dans le message envoyé. Sur l'exemple précédent, les ... renferment un grand nombre de propriétés comme la transparence de l'élément, les couleurs de fond et de texte, la taille de la police, etc. Plutôt que de ne pas renseigner ces valeurs lorsque celles-ci ne sont pas changées par l'utilisateur, elles sont renseignées dans tous les cas. L'idéal serait d'envoyer le minimum d'informations nécessaires. Lors d'une suppression, nous pourrions penser que l'ajout de tous les paramètres n'est pas nécessaire. Néanmoins, lors de l'annulation de l'action, il faut pouvoir rétablir les valeurs avant suppression.

Communication des changements

En observant l'activité du réseau dans le navigation, nous pouvons apercevoir comment les modifications sont envoyées. En effet, lors de cet envoi, un service REST intitulé *save* est appelé. Néanmoins, le procédé pour la réception du changement n'apparait pas.

Résumé

LucidChart envoie les modifications après chaque action d'un utilisateur au format JSON. Toutes les propriétés de l'élément modifié sont envoyées à chaque action dans le but de pouvoir annuler celle-ci.

Modélisation collaborative avec Papyrus

Phases supportées :

Description

L'initiative *Collaborative modeling initiative*[4] a été lancée grâce à un partenariat entre Eclipse, Obeo et le CEA et vise à permettre aux utilisateurs de collaborer au sein de Papyrus de manière asynchrone.

La figure 1.6 montre le fonctionnement de la collaboration asynchrone de Papyrus avec un modèle stocké sur un gestionnaire de sources comme Git.

Dans cet exemple, Cloé et Noémie travaillent toutes les deux sur un projet de bibliothèque du futur. Toutes les deux doivent maintenant ajouter de nouvelles fonctionnalités et doivent donc modifier le modèle. Cloé commence par créer une branche *localiserAmis* (étape 2). Noémie quant à elle crée une branche pour travailler sur la réservation des livres (étape 3). Après avoir réalisé les modifications nécessaires sur le modèle (étape 4), Cloé fusionne sa branche avec la branche principale *master* et envoie ses modifications sur le dépôt Git (étape 5). Tout se passe sans encombre puisqu'aucune modification n'a eu lieu sur le dépôt. Noémie quant à elle modifie le modèle (étape 6) et veut ensuite fusionner sa branche *réservationLivres* avec la branche principale *master*. Il lui est alors notifié qu'elle doit d'abord récupérer les changements effectués par Cloé. Seulement lors de la récupération de ceux-ci, des conflits apparaissent (étape 7). Celui-ci porte sur un même changement de multiplicité réalisé par Cloé et par Noémie. Cloé avait changé la multiplicité d'un élément avant d'envoyer ses modifications au dépôt central. Noémie ayant changé la multiplicité du même élément sans avoir conscience de changements doit alors choisir laquelle des deux modifications elle souhaite garder. Après avoir fait ce choix, elle peut alors envoyer ses modifications sur le dépôt Git (étape 8).

Lors de **la gestion d'un conflit**, l'utilisateur peut voir les changements réalisés comme le montre la figure 1.7. Sur cette figure, dans la partie basse, l'interface est divisée en 2 parties, montrant le changement

4. http ://eclipsesource.com/blogs/2015/04/13/collaborative-modeling-with-papyrus-emf-compare-and-egit/

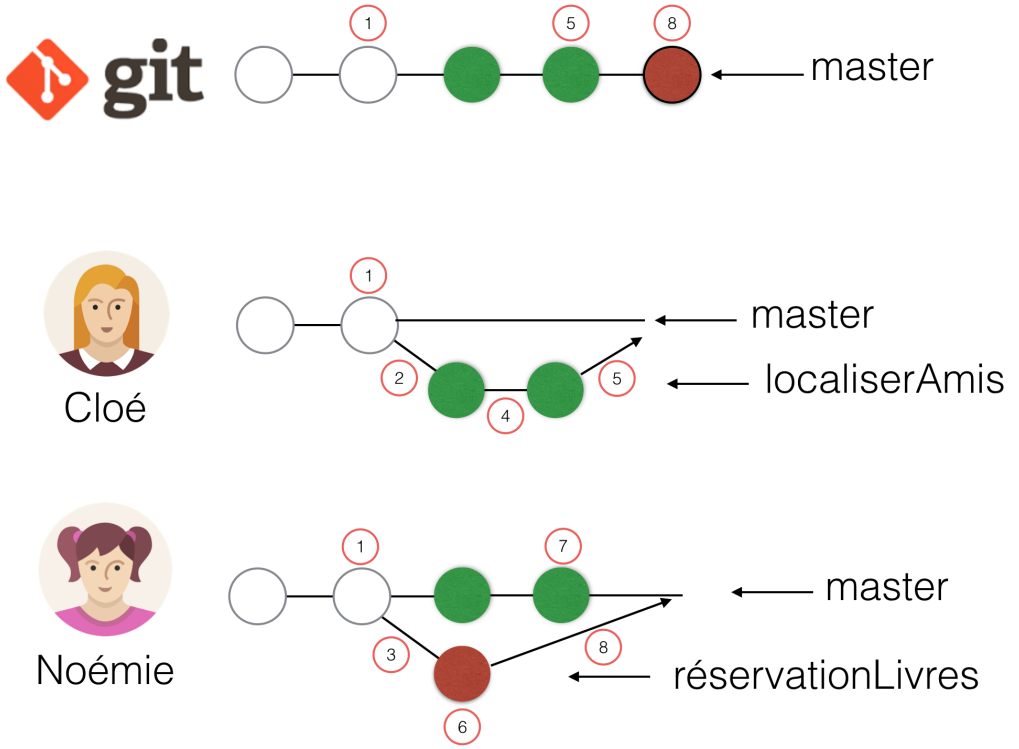


FIGURE 1.6 – Fonctionnement de la collaboration au travers d'un gestionnaire de sources

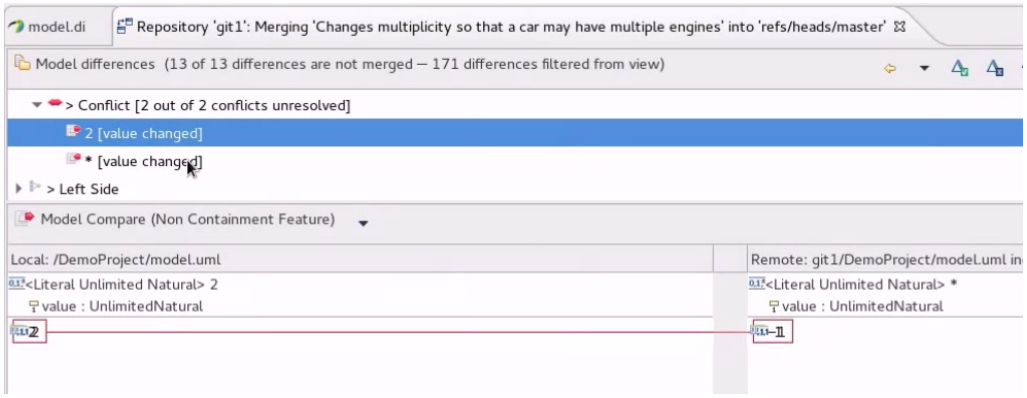


FIGURE 1.7 – Gestion d'un conflit dans la version collaborative de Papyrus

effectué par Noémie en local et la modification réalisée par Cloé, présente sur le dépôt distant. Noémie doit donc régler le conflit en choisissant laquelle des deux versions elle souhaite garder.

Stockage du modèle

En utilisant le gestionnaire de sources Git, Papyrus délègue le stockage du modèle.

Récupération du modèle

Git permet la récupération des données, ici du modèle, en fournissant des commandes shell et en particulier une nommée *pull*. Les données récupérées seront un fichier au format XMI, représentant le modèle.

Représentation des changements

Concernant **la représentation des changements**, c'est une nouvelle fois Git qui s'en charge. Pour chaque fichier modifié, Git va créer une nouvelle version de ce fichier et en changer le nom en le cryptant de manière à en faire un nom unique. Le commit comprend tous les fichiers ainsi créés. Le problème majeur de cette solution est que la taille du dépôt (en espace disque) grimpe très rapidement du fait de la duplication des fichiers. Par rapport à l'exemple présenté sur la figure 1.4, Git enverra le fichier complet, représenté en XMI. Le contenu du fichier sera le suivant :

```
<uml:Model xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="
    http://www.w3.org/2001/XMLSchema-instance" xmlns:uml="
    http://www.eclipse.org/uml2/4.0.0/UML" xmi:version="2.0
    " xmi:id="model_id" name="model">
<packagedElement xsi:type="uml:Package" xmi:id="
    package1_id" name="p1">
  <packagedElement xsi:type="uml:Class" xmi:id="class2_id"
      name="c2">
  </packagedElement>
</packagedElement>
<packagedElement xsi:type="uml:Class" xmi:id="class3_id"
    name="c3">
</packagedElement>
</uml:Model>
```

Communication des changements

Concernant **la communication des changements**, Git fournit plusieurs protocoles pour échanger les données (les commits comprenant

les fichiers complets ainsi que la réception de ceux-ci) entre les clients et le serveur : http, ssh ou git(similaire à ssh mais sans authentification).

Gestion des conflits

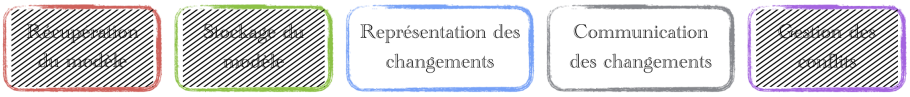
Afin de réaliser la différence de versions et de montrer aux utilisateurs ces changements lors de l'apparition de conflits, Papyrus utilise EMF Compare [5].

Résumé

Papyrus couvre toutes les phases de collaboration, en particulier grâce à l'utilisation de Git. Néanmoins, celle-ci n'est pas sans contraintes avec la taille du dépôt qui n'est pas optimisée. Enfin, la gestion des conflits peut s'avérer laborieuse pour les utilisateurs. L'obligation de la gestion des conflits étant due à la modélisation asynchrone.

Google Drawings

Phases supportées :



Description

Google fait partie des gros acteurs qui ont fait le choix de proposer de l'édition de documents collaboratives en ligne avec une suite intitulée Google Docs [6]. Elle permet d'éditer les mêmes documents de manière synchrone. J'ai décidé de présenter l'outil Drawings de Google puisque cet outil est utilisé par des millions d'utilisateurs et il s'agit de l'outil de la suite se rapprochant le plus de la modélisation.

Récupération du "modèle"

La représentation du dessin dans l'éditeur est faite par du SVG. Je suppose donc que lorsque l'utilisateur *récupère le dessin*, il reçoit le SVG directement. Néanmoins, cette étape est masquée et il n'est pas possible de déterminer précisément comment se fait cette récupération.

5. https ://www.eclipse.org/emf/compare/
6. https ://drive.google.com

Stockage du "modèle"

Le stockage des dessins est bien entendu effectué sur les serveurs Google. La manière de les stocker n'est par contre pas connu. Ceux-ci pourraient être stockés directement en base de données ou au travers d'un système de fichiers.

Représentation des changements

Lorsque les utilisateurs réalisent des modifications, ils échangent du JSON, **représentant l'action effectuée**. En observant les échanges réseau comme pour LucidChart, il est possible de voir la notation choisie pour les messages dont le format est le JSON. Lors de la création d'un nouvel élément, le message envoyé est le suivant :

```
[{"commands":[[TypeAction,"IdentifiantDeLElement",
   TypeElement,[longueur,?,?,hauteur,xPosition,
   yPosition],[?,"couleurBordure",?,"couleurFond",?,
   ?],"p"]],"sid":"TokenUtilisateur","reqId":
   NumeroDeLActionDansLaSessionCourante}]
```

Ce message contient les informations relatives au type d'action (ici une création) avec un identifiant attribué à l'élément mais aussi le type de l'élément, sa longueur, sa hauteur et son positionnement, ou encore un token attribué à l'utilisateur et le numéro de la commande lors de la session en cours. Ce numéro est réinitialisé à chaque ouverture d'un projet durant lequel personne n'était connecté auparavant. Certains des paramètres passés n'ont pu être déterminés.

La suppression d'un élément quant à elle se présente sous la forme :

```
[{"commands":[[TypeAction,["IdentifiantDeLElement"]]
   ],"sid":"IdentifiantDuProjet","reqId":
   TokenUtilisateur}]
```

Elle s'avère très simple puisqu'elle ne contient que le type de l'action, l'identifiant de l'élément à supprimer, le token attribué à l'utilisateur et le numéro de la commande lors de la session en cours.

Communication des changements

L'envoi de la commande au serveur se fait par un appel de service REST nommé *save*. Par contre, je n'ai pas pu déterminer comment les messages sont réceptionnés par les clients.

Résumé

Tout comme LucidChart, Google réalise l'échange de messages au format JSON en les envoyant par un service REST (REpresentational State Transfer). Les services REST permettent d'exposer des fonctionnalités ou des données à des partenaires extérieurs[Loreto et al., 2011]. La structure des messages adopté par Google est similaire à ceux de LucidChart. Néanmoins, ceux-ci ne contiennent que les informations nécessaires pour réaliser l'action contrairement à LucidChart.

1.6 Synthèse

La figure 1.8 synthétise le support des différentes étapes de la collaboration par les travaux et les outils présentés précédemment.

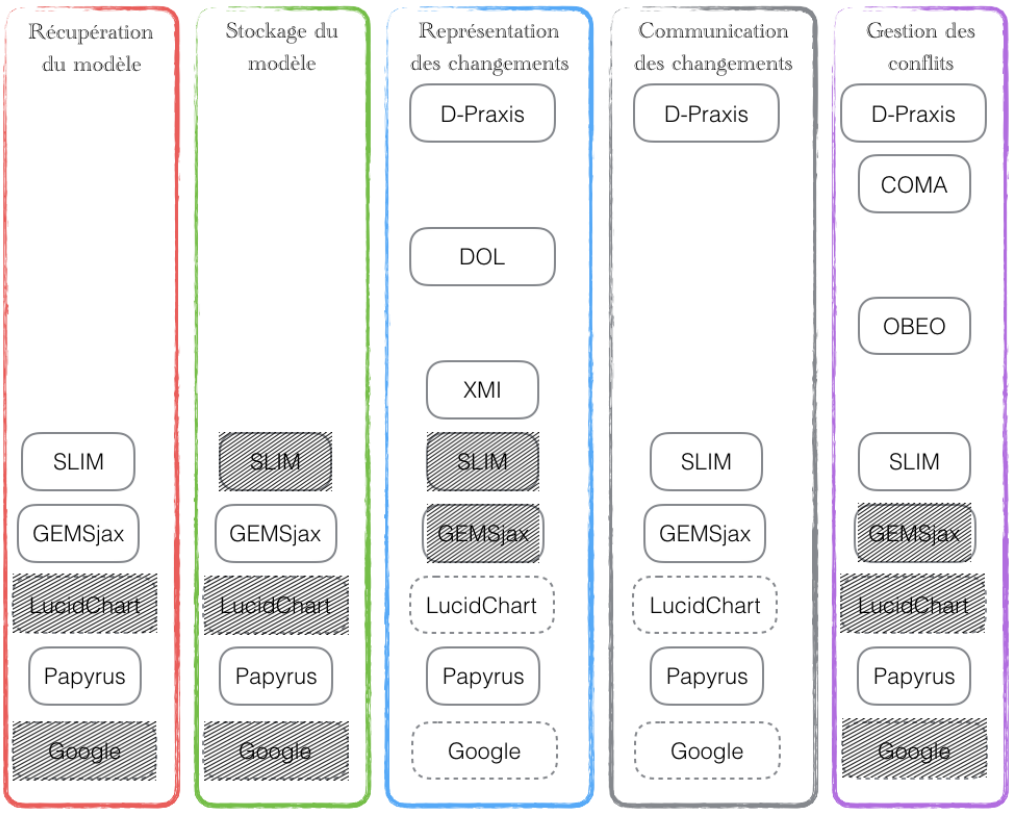


Figure 1.8 – Résumé du support des différentes phases de la collaboration

Tous ces travaux couvrent l'intégralité des phases de la collaboration. Pour la représentation des changements, les solutions de Google et de LucidChart sont valables à grande échelle mais ne se rapportent pas à la modélisation.

Le verrou actuel consiste donc à fournir une solution couvrant toutes les étapes de la collaboration au sein des outils de modélisation tout en garantissant une facilité d'ajout de nouveaux messages pour la mise en place de nouvelles fonctionnalités. Pour illustrer cette nécessité, prenons le cas de l'ajout d'une nouvelle fonctionnalité pour les utilisateurs : le réagencement des éléments graphiques du modèle. Pour la réaliser, une nouvelle action serait ajoutée permettant de changer le positionnement de ces éléments. L'ajout de cette nouvelle action ne doit pas impacter le système mis en place.

Un autre point important concernant l'échange des messages réside dans la généricité de la solution. L'élément commun aux outils de modélisation est la présence d'un méta-modèle. Le système se devra donc d'être valable quel que soit le méta-modèle utilisé.

Concernant le format des messages à échanger, la solution mise en place par Google a fait ses preuves en étant utilisée par des millions d'utilisateurs. Celle-ci devra être adaptée pour pouvoir s'appliquer à la modélisation. Tout d'abord, les utilisateurs de Drawings ne manipulent qu'un seul document. Lors d'une activité de modélisation, plusieurs diagrammes sont impliqués. De plus, le nombre d'actions réalisable est beaucoup plus important et chacune se doit de conserver la validité du modèle. Lors de la mise en place de la solution, je devrais tenir compte de ces différentes contraintes.

2 dAMOCleS : Système collaboratif

Mon but est de prendre en charge l'intégralité des phases de la collaboration. La solution mise en place se basera sur les observations réalisées lors de l'état de l'art. Nous nous baserons sur la grammaire adoptée par Google et LucidChart pour la représentation des changements en échangeant des messages au format JSON, adaptés à la modélisation. Le défi sera de faciliter la création de ces commandes JSON à l'instar de D-Praxis. De la même façon, la mise en place d'une collaboration synchrone permettra aux utilisateurs de se défaire de la gestion des conflits. Néanmoins, le système se devra de gérer les problèmes de synchronisation pouvant survenir. dAMOCleS (collAborative MOdeling Command System) est la solution mise en place que je vais présenter maintenant. Ce chapitre va présenter comment a été réalisé le support de chacune des phases de la collaboration avant de décrire comment s'est effectué le passage à l'échelle du système dans l'outil de modélisation GenMyModel.

2.1 Récupération et stockage du modèle

Lorsque l'utilisateur rejoint un projet, une requête vers le serveur est envoyée pour récupérer le modèle. Cette requête est effectuée sur une API fournissant des service REST. Lorsque cette requête est effectuée côté serveur, il interroge la base de données pour obtenir le modèle, stocké dans un format binaire. Le serveur renvoie ensuite ce binaire au client, dans le même format. Lorsque le client récupère le modèle au format binaire, il le convertit en modèle UML.

Ce modèle UML peut ensuite être manipulé au travers de commandes grâce à la librairie EMF [1](Eclipse Modeling Framework). Le but d'EMF est de faciliter la création d'outils et d'applications en fournissant des fonctionnalités d'édition et de visualisation des modèle. EMF facilite également la génération du code à partir du modèle. L'édition de modèle peut être effectuée à partir de commandes. Ces commandes de base sont les suivantes :

Add Ajoute un élément de modèle dans un conteneur. Exemple : ajout d'une classe dans un package.

1. https ://eclipse.org/modeling/emf/

Delete Supprime un élément de modèle avec toutes les références associées. Exemple : suppression d'une classe et des liens d'héritage s'y référant.

Move Déplace un élément de modèle dans la liste d'un conteneur. Exemple : changement de position d'un attribut dans une classe.

Remove Supprime un élément de modèle dans la liste d'un conteneur. Exemple : suppression d'un attribut d'une classe.

Set Change la valeur d'une propriété d'un élément de modèle. Exemple : change le nom d'une classe.

Compound Sert à encapsuler plusieurs commandes primitives.

Comme nous pouvons le voir, EMF ne propose pas de commande pour la création des éléments. Ce sont ces commandes qui seront exécutées au final sur le modèle. Il faut d'abord pouvoir faire transiter ces commandes sur le réseau. Pour cela, nous allons créer nos propres commandes, basées sur celle d'EMF.

2.2 Représentation des changements

La première étape commence par le clic de l'utilisateur que ce soit pour créer un élément de modèle, changer le nom d'une classe, ou pour résumer, chaque action que l'utilisateur peut réaliser sur le modèle. Cette action est ensuite convertie dans ce que l'on a appelé une commande, chaque commande permettant de faciliter la création des messages JSON.

Chaque action de l'utilisateur est donc décrite par un ensemble de commandes comme pour EMF. Néanmoins, certaines commandes ont été ajoutées, notamment pour les modifications graphiques pour en faciliter la création. Afin de pouvoir utiliser EMF pour la partie graphique, les représentations graphiques des éléments de modèle sont définies au travers d'un méta-modèle. Les modifications graphiques possibles, n'altérant pas le modèle sont les redimensionnements, les déplacements (changement de coordonnées dans le plan graphique), l'alignement d'éléments ou leur répartition. L'ensemble de ces commandes pouvant être réalisé sur un modèle et sa représentation graphique est présenté sur la figure 2.1.

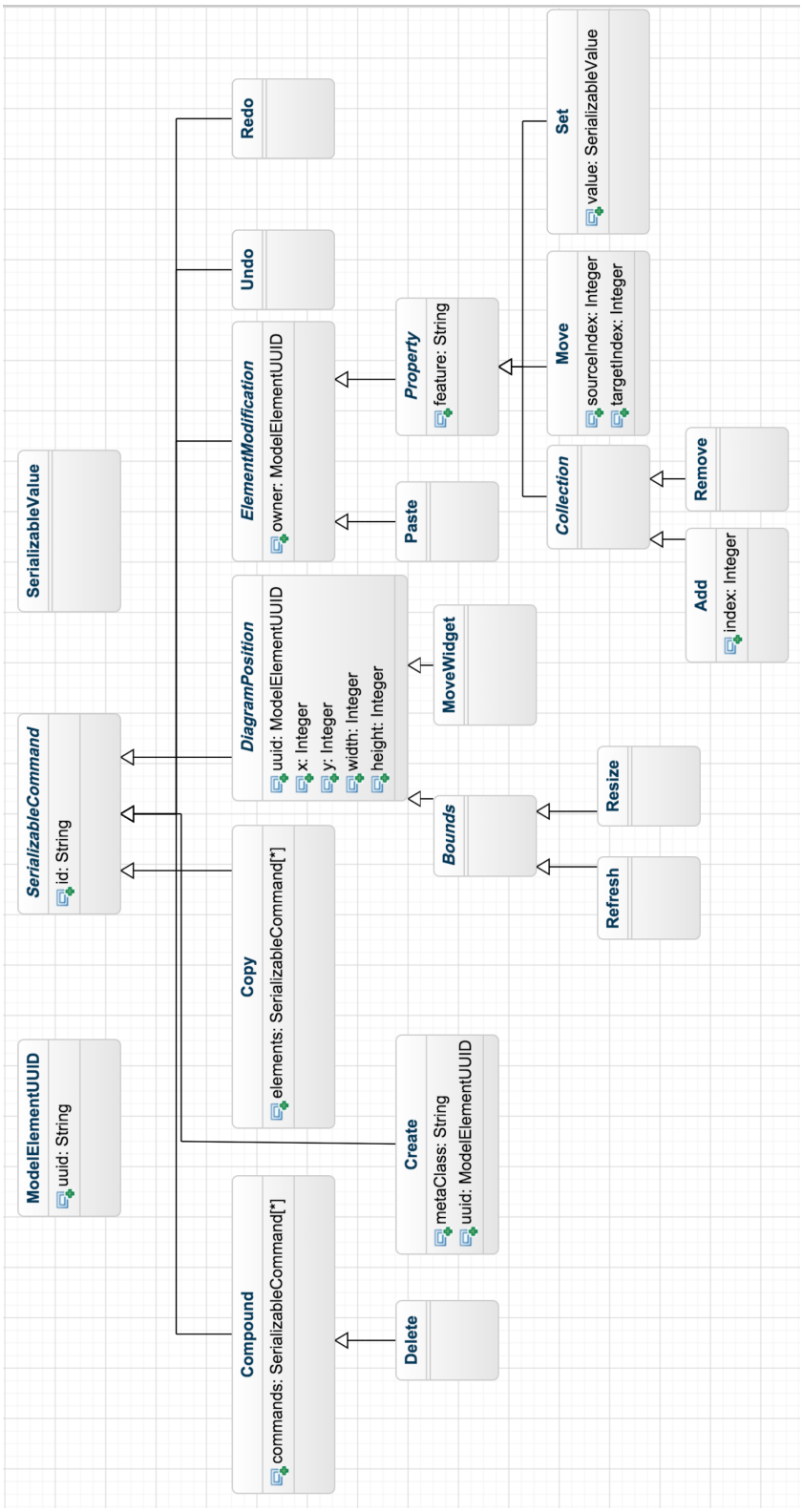


FIGURE 2.1 – Modèle représentant les commandes sérialisables

Ces commandes se basent sur D-Praxis, Google et LucidChart. Comme il s'agit d'une architecture répartie, les éléments de modèle se doivent d'être identifiés au travers d'identifiants uniques (UUID) et sont attribués au moment de leur création. Pour cette même création, la metaclasse est nécessaire pour indiquer le type d'élément à créer comme une Classe, un Package... De cette manière, la création d'éléments est générique. EMF permet de modifier les propriétés des éléments de modèles au travers de *feature*. Par exemple, lorsque l'on veut changer le nom d'une classe, la feature *name* sera utilisée. Dans les commandes personnalisées, cette même notion apparait (l'attribut feature dans la classe *Property*). Les commandes de base ont donc la définition suivante (* représentant un ensemble) :

- Compound(commandes*)
- Create(metaClass, uuid)
- Add(uuid, feature, uuidConteneur)
- Set(feature, uuid, valeur)
- Delete(uuid*)
- Move(feature, uuid, sourceIndex, targetIndex)
- Remove(feature, uuidParent, uuids*)

Voyons maintenant dans le détail la composition de la commande pour créer une classe, action la plus courante dans les outils de modélisation. Lors de la création d'une classe, il faut créer l'élément de modèle *Class* ainsi que sa représentation graphique appelé *ClassWidget*. Il faut ensuite attribuer un nom à cet élément de modèle, l'ajouter au modèle, changer le positionnement graphique de la représentation de la classe... Plutôt que de créer une seule commande contenant toutes les informations comme pour Google ou LucidChart. J'ai fait le choix de créer une commande contenant plusieurs micro commandes, chacune ayant une fonction précise.

Le verrou de départ était de pouvoir facilement ajouter de nouvelles fonctionnalités sans remettre en cause tout le système. En permettant la composition de commandes, il est ainsi possible de créer des commandes de haut niveau à partir des commandes de bases.

Lors du clic de l'utilisateur pour créer une classe, la commande créée est de la forme :

```
Compound
  Create("Class","uuidClass")
  Add("uuidClass", "packagedElement", "uuidModele")
```

```
Set("name","uuidClass", "MyClass")
Create("ClassWidget", "uuidClassWidget")
Set("modelElement", "uuidClass", "uuidClassWidget")
Set("x", "uuidClassWidget", 50)
Set("y", "uuidClassWidget", 50)
Add("ownedDiagramElement", "uuidClassWidget", "
   uuidModeleGraphique"
```

L'ordre des commandes est important. En effet, il ne sera par exemple pas possible de changer le nom d'une classe avant de l'avoir créée puisque les sous-commandes seront executées les unes après les autres. Une fois la commande construite, elle est convertie en JSON pour pouvoir être envoyée au serveur. Le format JSON a été choisi puisque celui-ci est un format standard pour le web et a fait ses preuves pour des applications de grande envergure. Le JSON est construit à partir de la commande en copiant chaque propriété ainsi que celles des sous-commandes. Le JSON contient donc le type de la commande et chacune des propriétés. Dans notre exemple, le JSON envoyé est le suivant :

```
{
"links": [],
  "commands": [
    {
    // commande contenant toutes les commandes
    "kind": "compound",
    "links": [
      {
      // projet sur lequel la commande doit Ãªtre executee
      "rel": "project",
      "href": "https://api.genmymodel.com/projects/
         identifiant_projet"
      },
      {
      // l'auteur de la commande
      "rel": "author",
      "href": "https://api.genmymodel.com/users/micheld"
      }
    ],
    // identifiant de la commande
    "commandId": 1,
    // date de creation de la commande
    "date": 1449051535870,
    // liste des commandes
    "commands": [
```

```
    {
      // creation d'une classe avec l'uuid uuidClass
      "kind": "create",
      "links": [],
      "uuid": "uuidClass",
      "metaClassName": "Class"
    },
    {
      // ajout de la classe creee dans un modele ayant
         pour uuid uuidModele
      "kind": "add",
      "links": [],
      "feature": "packagedElement",
      "objectId": {
        "uri": "genmymodel:uuidModele#/",
        "uuid": "uuidModele"
      },
      "values": [
      {
        "uri": "#//",
        "uuid": "uuidClass"
      }
      ]
    },
    {
      // change le nom de la classe ayant pour uuid
         uuidClass vers le nom MyClass
      "kind": "set",
      "links": [],
      "feature": "name",
      "objectId": {
        "uri": "#//",
        "uuid": "uuidClass"
      },
      "value": "MyClass"
    },
    {
      // creation d'un element graphique ClassWidget
         representant une classe avec comme uuid
         uuidClassWidget
      "kind": "create",
      "links": [],
      "uuid": "uuidClassWidget",
```

```
    "metaClassName": "ClassWidget"
  },
  {
    // lie l'element de modele ayant pour uuid
       uuidClass a sa representation graphique ayant
       pour uuid uuidClassWidget
    "kind": "set",
    "links": [],
    "feature": "modelElement",
    "objectId": {
      "uri": "#//",
      "uuid": "uuidClassWidget"
    },
    "value": {
      "uri": "#//",
      "uuid": "uuidClass"
    }
  },
  {
    // change la coordonnee x de l'element graphique
    "kind": "set",
    "links": [],
    "feature": "x",
    "objectId": {
      "uri": "#//",
      "uuid": "uuidClassWidget"
    },
    "value": 50
  },
  {
    // change la coordonnee y de l'element graphique
    "kind": "set",
    "links": [],
    "feature": "y",
    "objectId": {
      "uri": "#//",
      "uuid": "uuidClassWidget"
    },
    "value": 50
  },
  {
    // ajoute l'element graphique a la vue graphique
       du modele ayant pour uuid uuidModeleGraphique
```

```
        "kind": "add",
        "links": [],
        "feature": "ownedDiagramElements",
        "objectId": {
          "uri": "genmymodel:uuidModelGraphique#//%
             genmymodel%/@contents.0/@plane",
          "uuid": "uuidModeleGraphique"
        },
        "values": [
        {
          "uri": "#//",
          "uuid": "uuidClassWidget"
        }
        ]
      }
    ]
    }
  ]
}
```

L'action de départ de l'utilisateur étant maintenant convertie au format JSON, prête à être envoyée au serveur, nous pouvons maintenant voir par quel moyen ces commandes sont envoyées.

2.3 Communication des changements

Au tout début de la mise en place de la collaboration, j'ai utilisé une implémentation de CometD : Atmosphere [2] . Celle-ci permet d'adapter automatiquement le protocole utilisé en fonction du navigateur de l'utilisateur. Par exemple, si l'utilisateur possède un navigateur récent, le protocole utilisé sera des *WebSockets* de HTML5. Les WebSockets permettent l'échange d'informations bi-directionnelles entre un client et un serveur[Fette and Melnikov, 2011]. A l'inverse, si le navigateur est ancien, des requêtes AJAX[Garrett et al., 2005] sont employées. Celles-ci interrogent le serveur pour récupérer des informations. En utilisant AJAX, le client interroge donc à intervalle régulier le serveur pour obtenir ces informations. L'utilisation de ce framework s'est avérée être un échec. Lors d'un premier test réalisé avec des étudiants, la collaboration ne fonctionnait pas. En effet, un pare-feu mis en place par l'université bloquait l'échange de messages. Si les messages ne passent pas les pare-feu des universités, il y avait donc peu de chances que ceux-ci passent les

2. http ://async-io.org/

pare-feu mis en place dans certaines entreprises. De plus, les étudiants étant des utilisateurs majoritaires, une autre solution devait être trouvée.

Une solution alternative mise en place est Comet4GWT [3]. Pour l'échange de messages, des bus d'événements sont utilisés permettant ainsi de pouvoir séparer les types de messages facilement. Nous verrons plus tard que différents bus ont été mis en place, en plus de celui pour les messages concernant les actions des utilisateurs. Des *rooms* sont définies pour regrouper les messages pour certains utilisateurs en particulier. Dans le cas ici, une room est définie pour chaque projet en cours d'utilisation. La figure 2.2 illustre par un exemple le fonctionnement des rooms et du bus. Dans cet exemple, deux utilisateurs envoient une commande sur le *bus* nommé *command*. Lorsque les commandes sont réceptionnées sur le serveur, elles sont envoyées aux rooms correspondantes. Chaque room se charge alors d'effectuer les changements sur le modèle qui lui est lié.

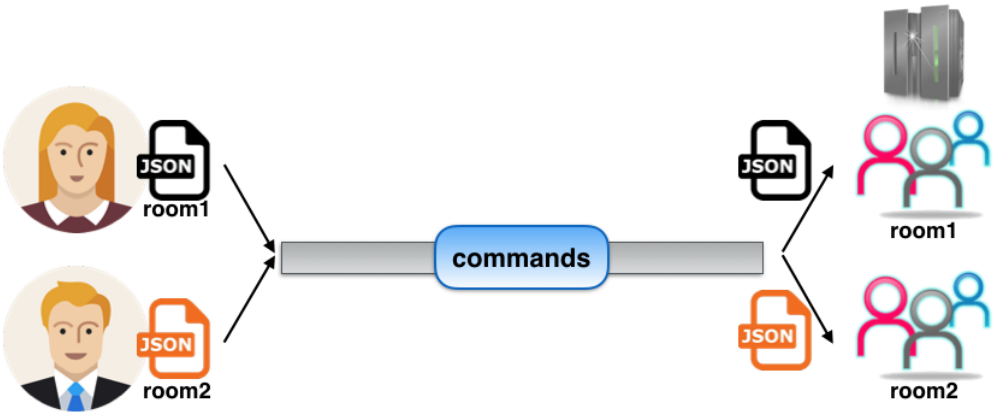


FIGURE 2.2 – Fonctionnement des rooms et du bus pour l'échange de commandes

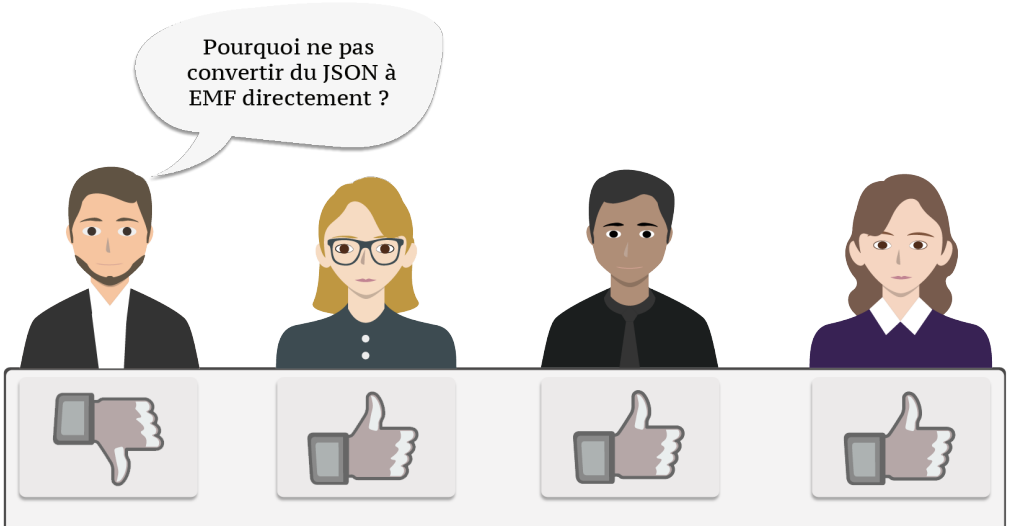
Avec CometD, la gestion des rooms se fait avec une classe Java contenant une méthode de réception et d'envoi par bus avec comme paramètre l'identifiant de la room.

Une fois que les actions ont été converties au format JSON, celles-ci sont donc envoyées sur le bus.

Le serveur réceptionne le message au format JSON au travers du bus pour ensuite convertir le en une commande. Ce nouveau passage vers une commande personnalisée n'est nécessaire qu'à des fins de sauvegardes de version. En effet, il est possible de sauvegarder ces commandes

3. https ://code.google.com/archive/p/cometd4gwt/

en base de données et de versionner les projets sur des commandes précises. Ces commandes sont ensuite converties vers des commandes EMF. Les commandes étant basées sur les commandes EMF, la conversion est très simple. La seule difficulté est de retrouver les éléments de modèles à partir des identifiants uniques. Pour cela, un parcours du modèle est réalisé pour retrouver l'élément. Afin d'optimiser la recherche, elle peut également se faire à partir du chemin de l'élément (élément uri visible dans le json), celui-ci étant construit à partir des différents uuid des éléments le contenant.
Ensuite, les propriétés représentées sous forme de chaines de caractères dans la commande peuvent être facilement converties vers des objets EMF.
La dernière étape côté serveur consiste à exécuter cette commande EMF sur le modèle. Pour cela, la commande contient la référence vers le projet sur lequel la commande doit être exécutée. A partir de cette référence, le modèle est chargé à partir de la base de données, de la même manière que le chargement du modèle lorsque l'utilisateur se connecte. Une fois chargé, la commande est exécutée sur le modèle.
Suite à l'exécution, dans le cas où il n'y a pas d'erreurs, le message au format JSON est renvoyé sur le bus pour parvenir à tous les clients, la commande, elle, est sauvegardée dans le base de données. La sauvegarde des commandes est nécessaire pour le undo/redo notamment. Les commandes de undo permettent d'annuler les dernières modifications et redo de les rejouer. Nous verrons également dans la suite de la thèse que les commandes sauvegardées seront utilisées pour une fonctionnalité collaborative.
Après envoi du message JSON sur le bus par le serveur, tous les utilisateurs connectés au projet vont le recevoir. Ils vont alors exécuter le même processus que le serveur. Le message JSON va d'abord être converti en une commande, puis vers une commande EMF pour ensuite s'exécuter sur le modèle. Les utilisateurs voient enfin la classe apparaitre !



Réponse : Comme dit plus haut, les commandes sont sauvées en base de données. Il n'est pas possible de sauvegarder les commandes EMF à la place. Le JSON étant du texte, il serait possible de le sauver en base. Néanmoins, il serait impossible d'effectuer des requêtes sur le JSON. De plus, certaines commandes EMF n'existant pas comme la création d'éléments ou des commandes graphiques complexes, il est plus facile de créer les commandes mises en place que du JSON directement.

2.4 Synthèse

La figure 2.3 résume le fonctionnement général du système dAMOCleS.

Lorsque l'utilisateur effectue une action comme une création de classe (1), une commande personnalisée est créée (2). Cette dernière est ensuite convertie dans un message au format JSON (3) avant d'être envoyée au serveur par le bus CometD (4). Après la réception, le serveur convertit le message JSON en une commande personnalisée (5), puis dans une commande EMF (6) pour pouvoir l'exécuter sur le modèle et stocker dans le même temps la commande personnalisée en base de données(7). Le message JSON reçu par le serveur à l'étape 4 est ensuite envoyé à tous les clients connectés au modèle par le biais du bus CometD. A la réception, les clients vont convertir, de la même manière que le serveur le message JSON en commande personnalisée (8), puis en commande EMF (9) pour l'exécuter sur le modèle local de l'utilisateur (10). Suite à cette exécution, l'utilisateur voit apparaitre la classe (11).

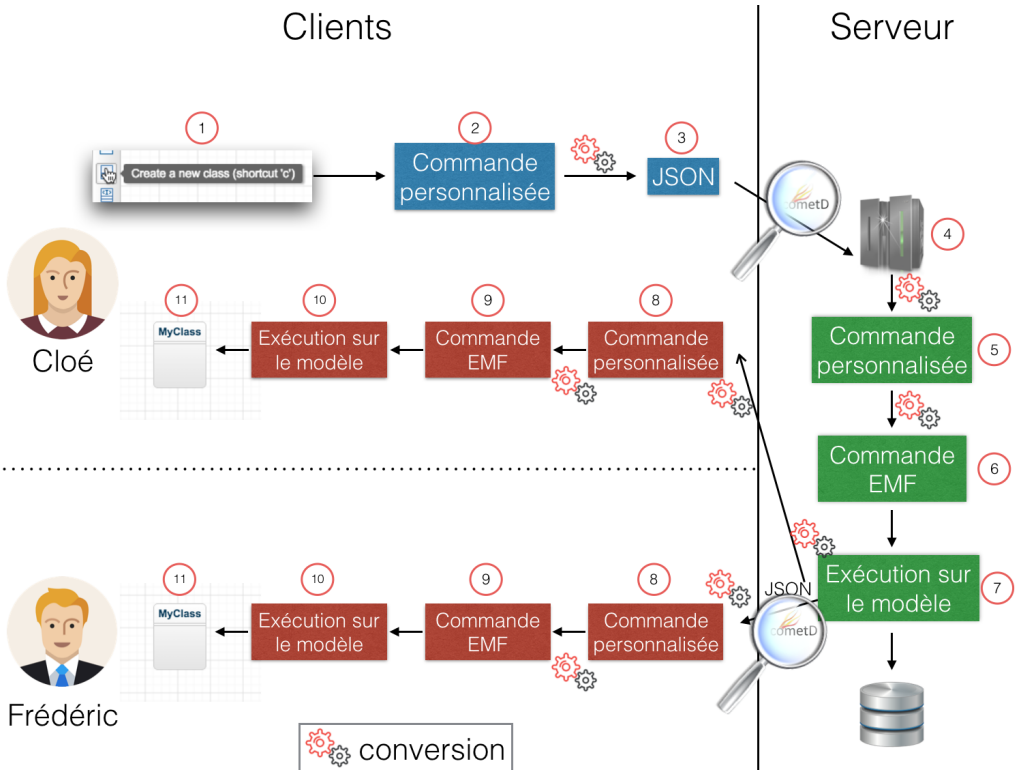


FIGURE 2.3 – Fonctionnement général du système de dAMOCleS

2.5 ACTIONS PARTICULIÈRES

Commande de suppression

La commande de suppression d'éléments nécessite un traitement un peu particulier. En effet, lorsqu'un élément est supprimé, il faut vérifier si d'autres éléments ont besoin d'être supprimés. La figure 2.4 montre un exemple de suppression d'élément impliquant d'autres suppressions. Sur cet exemple, la suppression de la classe *DiagramPosition* entraine la suppression de toutes les classes filles ainsi que du lien d'héritage avec la classe *SerializableCommand*. Les calculs déterminant les éléments à supprimer sont réalisés côté client pour ne pas surcharger la partie serveur. La commande de suppression contient donc plusieurs commandes de suppressions afin d'effacer les éléments n'ayant plus d'utilité.

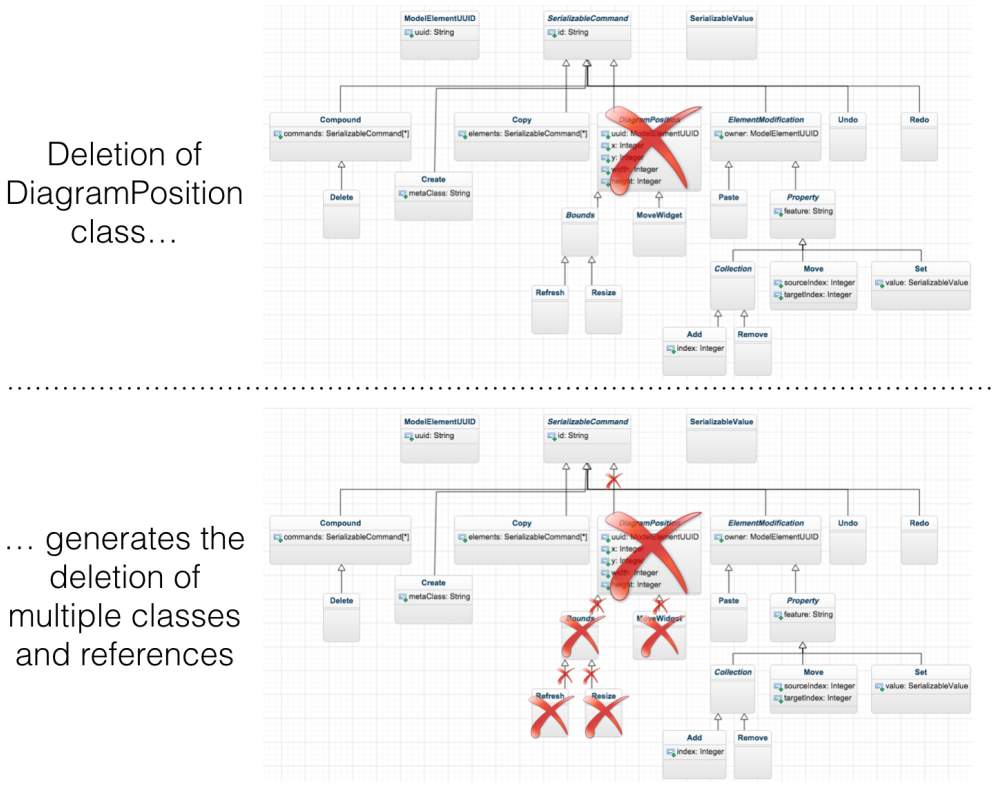


FIGURE 2.4 – Exemple de suppression d'élément

Commande de déplacement

Lorsqu'un élément de modèle est déplacé, le fonctionnement est légèrement différent. En effet, l'action est effective tout de suite côté client dans le but d'avoir un retour visuel immédiat. Lorsque l'utilisateur clique pour déplacer l'élément, il est obligatoire de déplacer celui-ci en même temps que le curseur de la souris. Si l'élément reste à sa place d'origine, l'utilisateur ne pourra pas le déplacer avec précision. De plus, un temps de latence aurait lieu durant le temps de l'aller-retour entre le client et le serveur. Ce temps de latence n'est pas gênant avec les autres actions. Néanmoins, en appliquant l'action avant l'envoi au serveur, une gestion d'erreur doit être prise en compte. Dès lors qu'une erreur se produit sur cette action, l'action est annulée.

2.6 Gestion des erreurs

Le choix d'une collaboration en temps-réel permet aux utilisateurs de ne pas à avoir à gérer les conflits comme vu lors de l'état de l'art. Néanmoins, certaines erreurs peuvent se produire que le système se doit de prendre en compte.

Validation de chaque action impossible

Contrairement aux outils de dessins comme LucidChart ou Cacoo, le modèle se doit d'être valide au sens UML. Ce qui signifie que certaines actions ne peuvent pas être réalisées par les utilisateurs. A titre d'exemple, les utilisateurs ne peuvent pas créer de lien d'héritage entre une classe et un package. L'IHM se doit donc d'intégrer des mécanismes de prévention, interdisant les actions invalides. Ne pas interdire les actions impossibles s'avérerait perturbant pour l'utilisateur si celles-ci sont annulées par la suite. Pour notifier l'utilisateur de ce type d'actions, une solution est de mettre en place des codes couleurs. Lorsque l'utilisateur veut réaliser une action conforme, elle apparait en vert, et en rouge dans le cas inverse. Pour revenir à l'exemple précédent, si l'utilisateur choisit de créer le lien à partir d'une classe, le package apparaitra en rouge pour lui notifier qu'il ne peut pas être la cible du lien.

Commande reçue dans l'intervalle de l'envoi de la "mienne"

Une autre erreur peut survenir dans le cas d'une modélisation distribuée dont au moins l'une des personnes possède une connexion lente. Elle peut se produire lorsqu'un utilisateur effectue une action mais reçoit une commande ne correspondant pas à la sienne. Pour palier ce problème, un système de compteur doit être mis en place. Lorsqu'un utilisateur envoie son action, un compteur est envoyé au travers de la commande. Ce compteur correspondant à celui de la commande précédente, incrémenté de 1. Lors de la réception d'une action, plusieurs cas sont possibles :

— Le compteur est bien celui attendu et la commande correspond à l'action de l'utilisateur. Dans ce cas, il n'y a pas d'erreur, il s'agit bien du comportement attendu.

— Le compteur est supérieur à la valeur attendu. Une ou plusieurs commandes n'ont pas été reçues. Le client est alors resynchronisé.

— Le compteur est bien celui attendu mais la commande ne correspond pas à l'action de l'utilisateur. Un autre utilisateur a donc déclen-

ché une action avant la sienne. Le client ayant reçu l'erreur est alors resynchronisé.

Le système présenté dans cette première partie de la thèse a été mis en place au sein de GenMyModel en Octobre 2013. Plus de deux ans après la sortie de la collaboration sur l'outil, je peux faire un retour d'expérience sur ce passage à l'échelle et présenter quelques chiffres pour montrer le succès de la solution.

2.7 Application à GenMyModel

Cette section présente maintenant l'intégration du système dAMOCleS au sein de l'outil GenMyModel.

Le protocole décrit précédemment et relatif aux commandes et aux échanges client-serveur est appliqué sur tous les projets des utilisateurs de GenMyModel, collaboratifs ou non, avec une petite différence lorsqu'un utilisateur travaille seul sur son projet. Dans ce cas, les actions sont d'abord exécutées sur le client avant d'être envoyées au serveur. En cas d'erreur, son action est annulée en le notifiant du motif.
Mi-Janvier 2016, environ 200 000 utilisateurs sont impliqués dans plus de 271 000 projets avec plus de 3 000 000 d'actions comme le montre le tableau 2.1. GenMyModel supporte 6 langages de modélisation différents : EMF, JPA, BPMN, FLOWCHART, UML et RDS [4], tous utilisant le système décrit dans cette partie. Le langage le plus utilisé est UML avec plus de 87% des projets dont 7,7% sont collaboratifs avec un nombre d'utilisateurs moyen par projet collaboratif de 2,8. Ces chiffres montrent une utilisation importante de la collaboration avec près de 20 000 projets.

Table 2.1 – Chiffres sur la mise en place de la collaboration (18 Janvier 2016)

	Projets		Projets collaboratifs			
	Nb	Nb actions	Nb	Nb actions	nb collaborateurs	Part projets
EMF	2050	26287	75	10886	2,5	3,7
JPA	495	15358	13	2471	2,6	2,6
BPMN	6930	256704	184	72068	2,4	2,7
FLOWCHART	9515	229455	295	55005	2,5	3,1
UML	237802	2112001	18220	974997	2,9	7,7
RDS	14765	530554	652	216024	2,6	4,4
total	271557	3170359	19439	1331451	2,9	7,16

4. https ://www.genmymodel.com/

L'importance du nombre d'actions effectuées, ainsi que le nombre de projets mis en place permet de dire que le système collaboratif est robuste pour un passage à grande échelle et ne se limite pas seulement à UML. **La solution adoptée est donc viable.**

2.8 Conclusion

Le système collaboratif se devait de respecter 2 critères :

l'ajout de nouvelles fonctionnalités pour l'utilisateur (et donc de nouveaux messages) ne doit pas impacter le système

le système doit s'appliquer sur n'importe quel méta-modèle

L'ajout de nouvelles fonctionnalités ne remet pas en cause le système grâce à la composition de commandes de base pour en faire des commandes de haut niveau. Par exemple, lors de l'ajout d'une nouvelle fonctionnalité permettant d'aligner des éléments de modèle, l'utilisateur n'aura à réaliser qu'une seule action, correspondant à une seule commande composée, comprenant plusieurs sous-commandes se chargeant de modifier les coordonnées de chaque élément.

De plus, en se basant sur le système de commandes EMF, servant à manipuler des modèles, le système est fonctionnel pour n'importe quel méta-modèle. Il a été appliqué aussi bien pour UML que pour BPMN ou d'autres types de modèles possédant un méta-modèle.

Résumé de la partie

La première partie de cette thèse a mis en valeur le moyen de mettre en place la communication entre différents clients dans les outils de modélisation. Après avoir fourni aux utilisateurs le moyen de collaborer, mes travaux se sont portés sur l'assistance à cette collaboration. Pour cela, des études sur l'utilisation de la collaboration et sur la définition des besoins ont été réalisés avant de fournir des réponses en terme de fonctionnalités collaboratives.

Deuxième partie

L'awareness pour une collaboration efficace

3 CSCW

Nous avons vu dans la partie précédente comment supporter toutes les étapes pour mettre en place la collaboration dans les outils de modélisation. Nous allons maintenant voir les concepts impliqués lorsque l'on parle de collaboration. Pour cela, nous allons d'abord nous poser la question de **qu'est ce que collaborer ?**

La définition admise pour la modélisation collaborative est :

> **La création conjointe de la représentation graphique partagée d'un système**[Renger et al., 2008]

Lors d'une collaboration, les personnes sont en interaction constante. Ceci nécessite qu'ils aient bien conscience qu'ils collaborent. Dans le cas d'un support informatique, cette conscience de la collaboration est possible si ce support fournit suffisamment d'informations la concernant comme *Qui fait quoi ?* ou *Qui travaille ?*. Ces informations constituent ce que l'on appelle la **collaboration awareness**. Nous détaillerons cette notion par la suite et nous la nommerons simplement *awareness*.

Depuis plus d'une trentaine d'années, une communauté scientifique s'est constituée autour de la problématique du travail collaboratif assisté par ordinateur *CSCW* (Computer-Supported Cooperative Work) dont le but est d'améliorer le travail collaboratif et la productivité grâce au support de la technologie[Carstensen and Schmidt, 1999]. Ce chapitre présente les composantes principales de la collaboration selon un modèle appelé 3C défini par Ellis et al. en 1991[Ellis et al., 1991] et étendu par Fuks[Fuks et al., 2005]. Les 3C désignant **communication**, **coordination** et **coopération**.

3.1 Modèle 3C

La **communication** se rapporte à la négociation et à la prise de décision[Fuks et al., 2005]. Lorsque les personnes se trouvent dans un lieu commun, la communication se fait au travers de plusieurs moyens comme se rendre au bureau de son collègue, discuter autour d'un café, les réunions, les "scrums meetings", les emails ou encore les messages instantanés-[Omoronyia et al., 2010].

La **coopération** concerne la production commune réalisée par les membres d'une équipe dans un environnement partagé, générant et manipulant des objets coopératifs dans le but d'accomplir des tâches. La **coordination** quant à elle est centrée sur la gestion des personnes, des activités et des ressources[Fuks et al., 2005].

Dans un monde parfait, nous pourrions nous passer de communiquer pendant l'accomplissement d'une tâche en planifiant tout le projet dès le départ, en répartissant les rôles à chacun, etc. En réalité, des événements imprévus surviennent comme des estimations inappropriées, des changements de technologies ou encore des changements de personnes dans les équipes[Herbsleb and Grinter, 1999]. Dans le cas où les équipes sont réparties, la communication entre elles se fait rare ce qui empêche la prise de conscience rapide de conflits sur des travaux communs. Bien souvent, certaines informations ne sont pas transmises entre les équipes comme la priorité des tâches à accomplir ou le niveau d'expertise des membres des équipes. La communication avec des équipes distribuées apporte d'autres difficultés avec les différences culturelles, politiques ou encore temporelles[Ivček and Grbac, 2008]. Les notions de communication, coordination et coopération sont en interaction constante [Gerosa et al., 2006]. En effet, il est nécessaire de transmettre ces informations entre les différentes équipes pour prendre des décisions durant cette coopération. Ceci engendre une communication entre les équipes, nécessitant une coordination pour réorganiser les tâches devant être effectuées durant cette coopération comme le montre la figure 3.1. Cette figure montre également la nature itérative de la collaboration. Les participants à cette collaboration obtiennent des retours à partir de leurs actions et de celles des autres membres au travers des informations d'awareness relatives aux interactions entre les participants. La notion d'awareness est la notion centrale de la collaboration, rendue plus difficile lorsque la collaboration se fait dans un environnement distribué [Cramton, 2001].

3.2 Awareness

La définition communément admise pour l'awareness est celle de Dourish[Dourish and Bellotti, 1992] :

> **an understanding of the activities of others, which provides a context for your own activity**

La notion de contexte est utilisée pour vérifier que les contributions individuelles sont pertinentes pour l'activité du groupe dans le but d'accomplir les tâches communes. Ces premiers travaux sur l'awareness, définis en 1992 avaient pour but d'aider à l'écriture de texte collaborative. Cette initiative a néanmoins été reprise très largement dans le domaine du génie logiciel. Dans ces deux cas, les composantes de l'awareness restent les

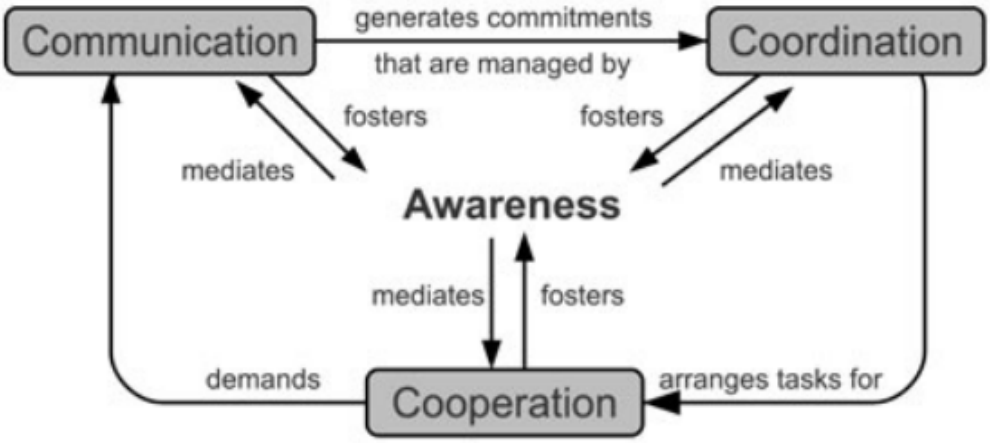


FIGURE 3.1 – Modèle 3C à partir de [Ellis et al., 1991], étendu par [Fuks et al., 2005]

mêmes. L'awareness est découpé en 4 dimensions[Omoronyia et al., 2010]. Le **Workspace**, l'**Informal**, le **Group-Structural** et le **Social** awareness.

Workspace awareness : Informations sur les interactions les plus récentes des utilisateurs avec l'environnement partagé[Gutwin et al., 1995].

Informal awareness : Informations générales sur les personnes participant au projet, ce qu'elles font et ce qu'elles prévoient de faire [Gutwin et al., 1996a].

Group-Structural awareness : Apporte une connaissance à propos du rôle des collaborateurs, leur place dans le projet, leur statut... [Gutwin et al., 1996a].

Social awareness : Donne des informations sur la présence et les activités des personnes[Prinz, 1999].

Lorsqu'une équipe travaille ensemble, chaque membre peut se poser des questions à propos des autres ou de l'environnement partagé comme *Avec qui suis-je en train de travailler ?*, *Sur quoi travaillent les autres ?*, *Que sont-ils susceptibles de modifier ?*. Toutes ces questions sont relatives à l'awareness. Bien que la définition de l'awareness soit clairement définie, les éléments le composant sont très souvent différents [Steinmacher et al., 2013] [Gutwin and Greenberg, 2002]. Cette différence vient généralement de la granularité donnée aux informations. Les questions que l'on peut se poser sont présentées dans le tableau 3.1. Chaque question appartient à l'une des dimensions présentées précédemment.

Le tableau présente également le nom général ayant été donné à chaque question. Les éléments d'awareness apparaissant ici sont présentés avec la granularité la plus fine.

TABLE 3.1 – L'awareness - Les questions qu'une personne peut se poser lorsqu'elle collabore [Steinmacher et al., 2013]

Dimension	Element	Question
Workspace	Identity	Qui est-ce ?
	Authorship	Qui fait ceci ?
	Action	Que font-ils ?
	Action history	Comment cette opération s'est-elle produite ?
	Intention	Dans quel but mon action s'inscrit-elle ?
	Intention history	Dans quel but mon action s'inscrivait-elle ?
	Artifact	Sur quels objets travaillent-ils ?
	Artifact history	Comment les objets sont-ils arrivés dans cet état ?
	Location	Où travaillent-ils ?
	Location history	Où ont-ils travaillé ?
	Gaze	Où regardent-ils ?
	View	Que peuvent-ils voir ?
	Reach	Où peuvent-ils intervenir ?
	Event history	Quand s'est produit cet événement ?
Informal	Opinion	Qu'en pensent-ils ?
	Presence	Une personne est-elle connectée ?
	Presence history	Qui était présent et quand ?
Social	Interest level	Quelle est leur implication ?
	Emotional feelings	Comment se sentent-ils ?
	Availability	Sont-ils disponibles ?
Group-structural	Roles and responsibilities	Quels sont leurs rôles/responsabilités ?

Workspace awareness

Le workspace awareness concerne les informations à propos des interactions des autres collaborateurs sur l'environnement partagé et sur les artéfacts qu'il contient. La définition admise par la communauté CSCW au workspace awareness est *who is working on what*[Vertegaal, 1997], définie par Vertegaal ou encore *...the up-to-the-minute knowledge of other participants' interactions with the shared workspace*[Gutwin et al., 1995] définie par Gutwin. Pour Gutwin, cet environnement partagé joue un rôle très important pendant la collaboration en maintenant la connaissance sur les interactions des autres sur l'environnement et les artéfacts qu'il contient[Gutwin et al., 1996a].

Informal awareness

La définition associée à l'informal awareness est *...the general sense of who is around, what are they doing, and what are they going to do*

[Gutwin et al., 1996a]. Il s'agit des informations que l'on peut obtenir à partir des discussions informelles lorsque des personnes partagent le même bureau[Gutwin et al., 1996a]. D'après la définition, certaines informations d'awareness sont partagées entre le workspace awareness et l'informal awareness comme *Identity* ou encore *Location* mais la communauté CSCW admet ces informations uniquement en workspace awareness. Dans certains cas, l'informal awareness est considéré comme une sous-partie du workspace awareness[Omoronyia et al., 2010] .Ces informations sont directes lorsque les membres d'une équipe partagent le même bureau. Néanmoins, dans le cas d'équipes distribuées, un support technologique est nécessaire pour fournir ces informations
[Dourish and Bellotti, 1992]. Le défi pour fournir ces informations est de savoir comment les capturer, les traiter et les afficher aux utilisateurs
[Gross et al., 2005]. L'information d'awareness ayant eu le plus gros support technologique est la présence au travers des messageries instantanées. Dans les messageries instantanées, les utilisateurs peuvent voir si les autres personnes sont connectées et si elles sont disponibles.

Group-structural awareness

La définition utilisée pour le group-structural awareness est : *...people's roles and responsibilities, their positions on an issue, their status, and group processes*[Gutwin et al., 1996a]. Dans les sciences de l'information, les rôles sont représentés comme des éléments indépendants d'un système et peuvent être définis séparément. Ces rôles peuvent ensuite être pris par un ensemble d'une ou plusieurs personnes en même temps
[Zhu, 2003][Genilloud and Wegmann, 2000]. Lors d'une activité collaborative, en ayant les rôles bien définis et attribués, une notion de hiérarchie et d'expertise apparait. Par exemple, si des rôles de *développeur*, *architecte*, *chef de projet* et *client* sont définis, le développeur sait qu'il doit s'adresser à l'architecte en cas de questions concernant le modèle. Le client quant à lui se référera au chef de projet.

Social awareness

La définition admise pour le social awareness est toujours celle définie par Gutwin : *...the understanding that participant's have about the social connections within their group*[Gutwin et al., 1995]. Fournir des informations comme l'intérêt ou l'état émotionnel[Gross et al., 2005] permet de minimiser les interruptions et perturbations en étant engagé dans un

processus collaboratif, ce que Schmidt définit comme *appropriate obtrusiveness*[Schmidt, 2002].

3.3 Lien entre les informations d'awareness et les fonctionnalités collaboratives

Dans l'un de leurs articles, Gutwin et Greenberg [Gutwin and Greenberg, 2002] ont fait le lien entre les informations d'awareness et une ou plusieurs façons de les représenter. Leurs travaux ont été repris à plusieurs reprises notamment par [Heinrich et al., 2013] pour fournir un framework graphique pour représenter les informations d'awareness, notamment pour le *Workspace Awareness*. Gutwin est l'un des chercheurs les plus actifs sur l'awareness. Ses propositions pour le support des fonctionnalités collaboratives relatives au *Workspace Awareness* vont être présentées maintenant. Ses travaux s'inscrivent dans le domaine de l'édition de documents collaborative, première activité étudiée pour la mise en place de l'awareness.

D'autres travaux s'ajouteront permettant le support des autres types d'awareness.

Presence

L'information de présence sert à identifier si d'autres collaborateurs sont connectés sur l'environnement partagé. Selon Gutwin, il s'agit de l'affichage d'information d'awareness le plus basique.

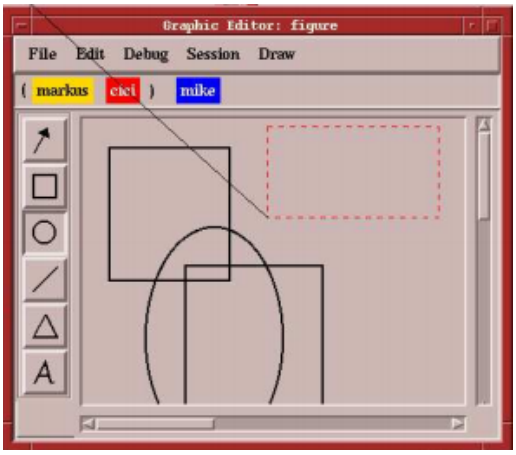


Figure 3.2 – Affichage des informations *Presence*, *Identity* et *Authorship* selon [Sohlenkamp and Chwelos, 1994] en 1994

En affichant une liste de participants comme inspiré par [Sohlenkamp and Chwelos, 1994], comme le montre la figure 3.2, les collaborateurs sont ainsi en mesure de voir rapidement s'ils sont seuls à être présents sur le projet ou non. Dès lors que l'utilisateur s'identifie dans le système, une notification est envoyée aux autres collaborateurs pour avertir de sa présence.

Identity

En complétant la liste des participants, il est possible de connaitre leur identité au travers d'une image les représentant ou de leur nom à partir de leur profil utilisateur. Comme sur la figure 3.2 présentant la liste des personnes connectées, le nom des utilisateurs apparait.

Authorship

Afin de connaitre l'auteur d'une action, les éléments manipulés par les utilisateurs sont colorés, généralement à partir de la couleur attribuée à un utilisateur dans la liste des utilisateurs connectés comme sur la figure 3.2.

Action

Afin de montrer l'activité de l'ensemble de l'équipe, il est possible de mettre en place des indicateurs d'activité montrant l'intensité des changements dans l'environnement. Il est également envisageable d'afficher les curseurs de souris de chaque collaborateur ou d'émettre des sons correspondant à différents types d'actions. Une autre solution est la mise en place d'animations lorsque les actions ont lieu de manière instantanée. La figure 3.3 montre un exemple d'affichage des curseurs de souris sur la partie E tout comme la figure 3.4.

Les travaux de Maitland et al.[Maitland et al., 2006] quant à eux, montrent une représentation possible du niveau d'activité physique de groupe. Ils enregistrent l'activité des personnes pour ensuite pouvoir les comparer les uns aux autres comme l'illustre la figure 3.5.

Des travaux ont également été réalisés dans le but de fournir une librairie graphique pour la représentation des informations d'awareness. GAwI [Heinrich et al., 2013] propose également une fonctionnalité de partage des curseurs de souris des utilisateurs comme le montre la figure 3.6 sur la partie a(I).

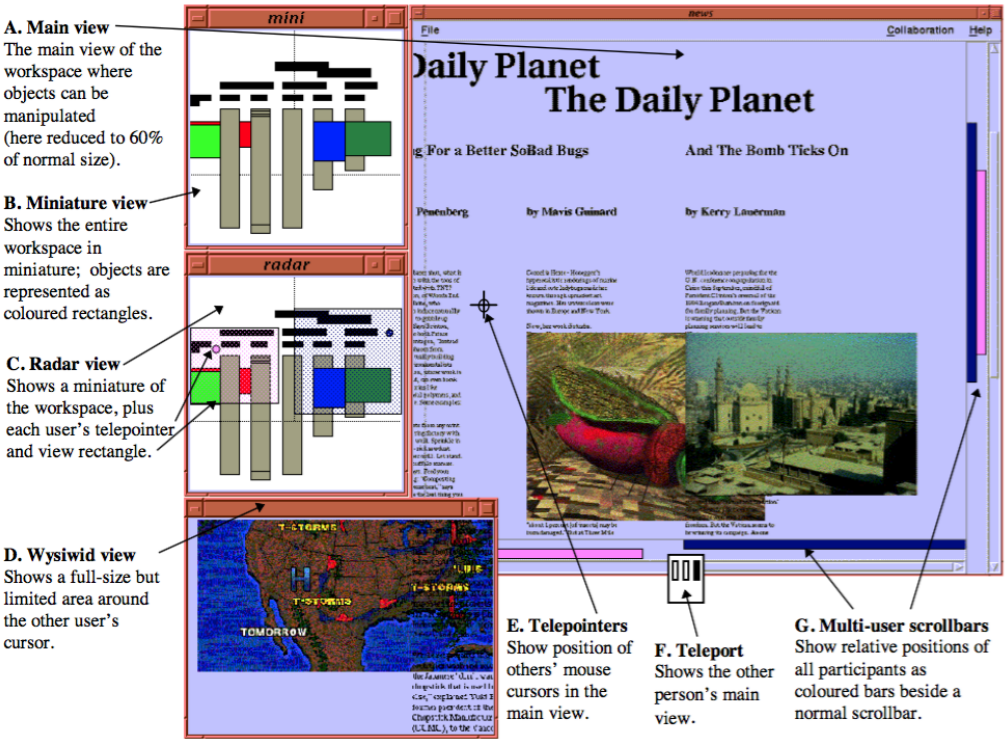


FIGURE 3.3 – Affichage des informations *Action*, *Artifact*, *Location*, *Gaze*, *View* et *Reach* selon [Gutwin et al., 1996b] en 1996

Action/Artifact History

L'outil Diffamation [Chevalier et al., 2010] permet d'identifier les changements réalisés entre deux versions d'un document.

La figure 3.7 montre la représentation des informations d'awareness. L'outil se découpe en 3 parties. La partie a représente la vue principale du document sur lequel apparaissent en vert les ajouts et en rouge les suppressions. Sur la partie b, la vue générale de l'intégralité du document est présentée. La partie c quant à elle indique la *timeline* des modifications. Cette représentation permet d'afficher les informations d'awareness *Action History* et *Artifact History*.

Intention

La mise en valeur en temps-réel des éléments manipulés permet de prédire les mouvements et d'anticiper les actions. A chaque fois qu'un utilisateur interagit avec un élément de l'environnement partagé, un évé-

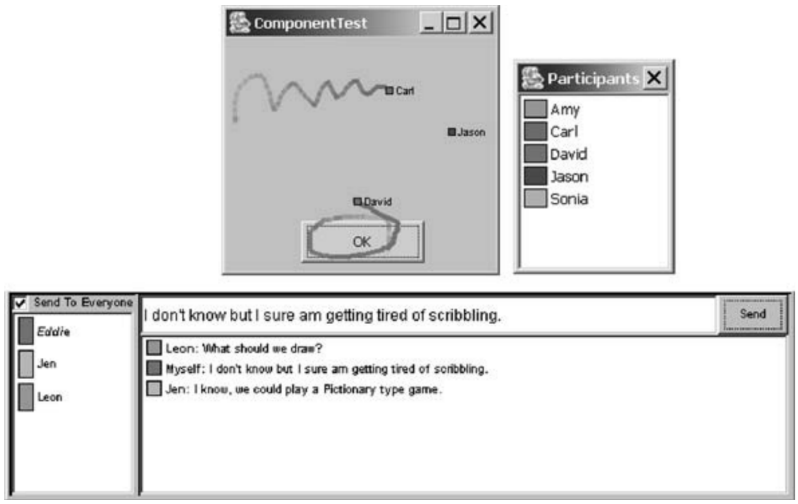


FIGURE 3.4 – Partage des pointeurs de souris selon [Hill and Gutwin, 2004] en 2004

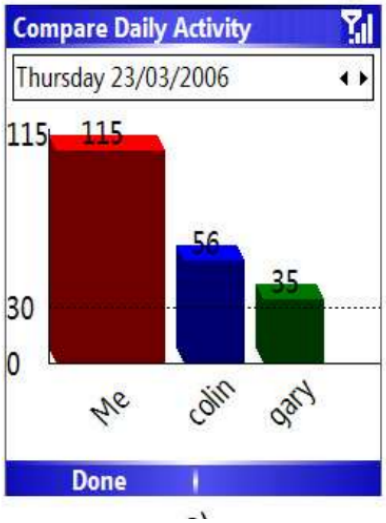


FIGURE 3.5 – Affichage du niveau d'activité selon [Maitland et al., 2006] en 2006

nement est envoyé à ses collaborateurs. GAwI parle également du support des intentions dans son framework. Pour fournir cette information, ils utilisent le pointeur de souris des utilisateurs comme l'illustre la figure 3.6 sur la partie b. Néanmoins, la fonctionnalité ne permet pas de prédire complètement les intentions des utilisateurs. En effet, lorsqu'une

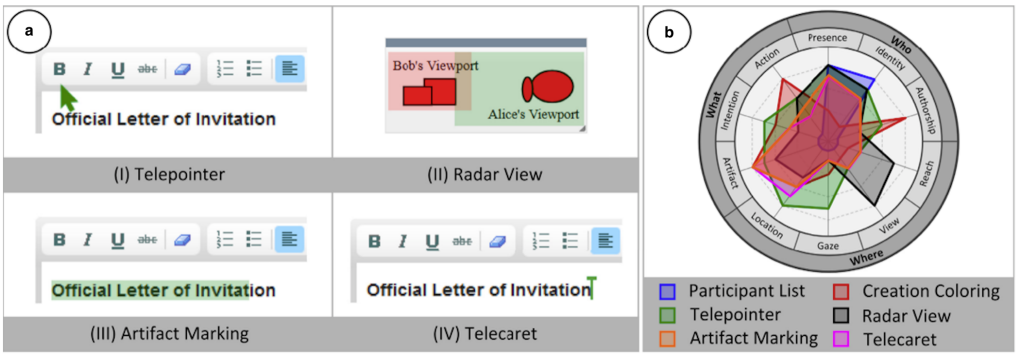


FIGURE 3.6 – Librairie GAwI pour représenter le *Workspace Awareness* en 2013 [Heinrich et al., 2013]

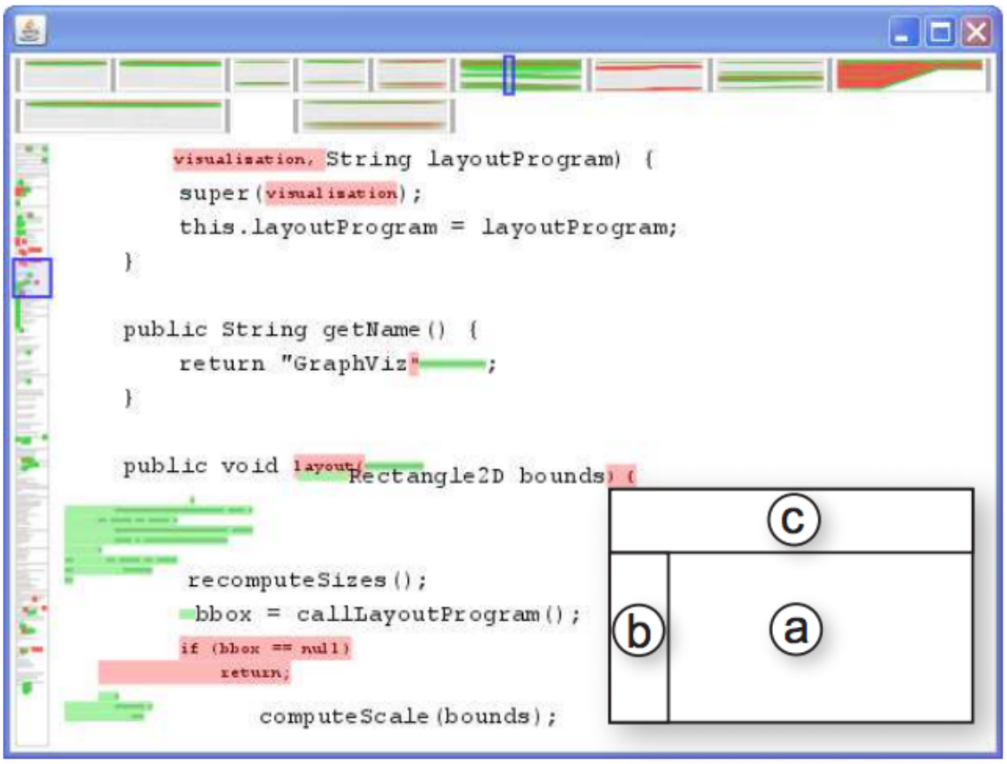


FIGURE 3.7 – L'outil Diffamation compare deux version d'un document et fournir les informations *Action History*, *Artifact History* et *Location History* [Heinrich et al., 2013]

personne se place sur une zone de texte, il n'est pas possible de savoir

si le but est d'ajouter du texte, le modifier ou le supprimer. Les intentions se rapportent également aux tâches. Pour ce faire Task Manager [Kreifelts et al., 1993] propose aux utilisateurs de les gérer au travers de leur outil.

Intention History

Task Manager permet également d'accéder à l'historique des tâches et de voir ainsi l'activité des utilisateurs depuis une date donnée.

Artifact

Les éléments manipulés peuvent être connus au moyen d'une liste à l'instar de la présence des participants. Une autre possibilité est une nouvelle fois l'émission de sons donnant des indications sur les éléments en cours de manipulation. GAwI propose de colorer les éléments manipulés par les autres. La figure 3.6 montre une nouvelle fois une fonctionnalité adressant l'information *Artifact* sur la partie a(III).

Location

À partir des coordonnées du pointeur de la souris d'un utilisateur, une vue radar affiche le positionnement des pointeurs ou des rectangles sur une miniature représentant l'environnement partagé. Les travaux existants font état de deux représentations :

Des scrollbars latérales, indiquant la partie du document que les utilisateurs peuvent voir comme montré sur la figure 3.3 sur la partie G.

GAwI quant à lui utilise une nouvelle fois le pointeur de souris des utilisateurs.

Location History

Les travaux sur l'outil Diffamation [Chevalier et al., 2010] permettent également de cibler l'information d'awareness *Location History*. En couplant la *timeline* avec les modifications réalisées, il est possible de voir sur quelles parties du document un utilisateur a travaillé.

Gaze

Pour obtenir l'information concernant la vision des utilisateurs, il est proposé d'utiliser des techniques d'*eye tracking*, suivant le mouvement

des yeux. C'est ce que proposent Vertegaal et al. [Vertegaal et al., 1997] sur la figure 3.8. Chaque utilisateur possède une couleur qui lui est attribuée, apparaissant sur les écrans et sur la feuille placée sur la table. Cette couleur est utilisée pour afficher des cercles colorés, correspondant au point de visualisation de l'utilisateur.

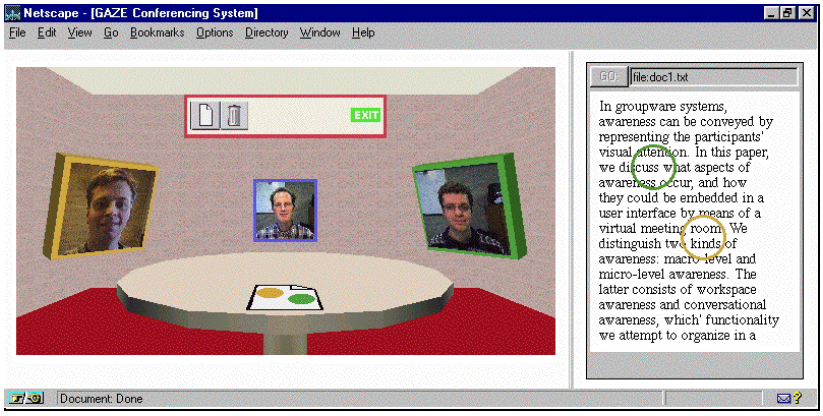


FIGURE 3.8 – Utilisation de l'eye tracking pour représenter l'information *Gaze* en 1997 [Vertegaal et al., 1997]

View

L'information sur ce que les utilisateurs peuvent voir est mise en place au travers de rectangles représentant ce qu'ils voient à l'écran. Ainsi, en ayant une vue globale de l'environnement partagé sur lequel les rectangles sont ajoutés, les zones sont affichées pour chaque utilisateur. Pour afficher ces rectangles, il faut envoyer à tous les utilisateurs la zone d'affichage dès lors que celle-ci est changée. Pour représenter cette informations, GAwI utilise une vue radar comme illustré sur la figure 3.6, partie a(II).

Reach

L'information *Reach* se rapporte à ce que les utilisateurs peuvent réaliser comme changements. Gutwin propose également de symboliser par des rectangles les zones visibles par les utilisateurs comme pour l'information *View*.

Opinion

L'*opinion* est très peu traitée dans la littérature. La plupart du temps, l'opinion apparait au travers des moyens de communications comme c'est la cas pour *The Open Meeting*[Hurwitz and Mallery, 1995]. Comme Tom Gross a pu le mentionner[Gross, 1997], l'*Informal Awareness*, dont fait partie l'*Opinion* est traité au travers de nouveaux moyens de communication, servant à palier le manque de communication en face-à-face. *The Open Meeting* est une plate-forme d'échange de discussions en ligne. Dans le but de donner leur opinion à chaque message, les utilisateurs répondent à chaque message en classifiant leur message selon les catégories suivantes :

Agree : L'utilisateur approuve l'action ou la recommandation.

Qualify : Explication des exceptions ou des extensions pour une recommandation ou une action.

Alternative : Une façon alternative d'implémenter une recommandation ou une action.

Disagree : Indique une façon de contester pourquoi ou comment une recommandation ou une action fonctionne.

Example : Rapporte une pratique prometteuse qui illustre une bonne façon de réaliser une recommandation.

Question : Pose une question à propos d'une recommandation ou d'une action.

Answer : Répond à la question de quelqu'un.

A chaque type de message est associée une icône. La figure 3.9 résume l'icône utilisée en fonction du type de message.

Presence History

Presence History se réfère aux événements de connexion et de déconnexion des utilisateurs. Un cas typique d'utilisation de cette information se trouve dans les messageries instantanées de groupe. Avant les années 2000, Viegas et Donath[Viégas and Donath, 1999] ont présenté un système de logs ajoutés dans la messagerie instantanée pour notifier des connexions et déconnexions comme l'illustre la figure 3.10.

Interest Level

Les travaux de Claypool et al.[Claypool et al., 2001] ne ciblent pas le CSCW. Néanmoins, ils pourraient s'y appliquer. Les auteurs ont défini

Icon	Link Type	Description
	Agree	A reason to support the recommendation or action.
Yes, but...	**Qualify**	A qualification that explains exceptions or extensions for a recommendation or action.
	Alternative	An alternative way to implement a recommendation or action.
	Disagree	A reason to challenge why or how a recommendation or action can work.
	Example	A report of a promising practice that illustrates one good way to realize a recommendation.
	Question	A question about a recommendation or action.
	Answer	An answer to someone else's question.

FIGURE 3.9 – Iconographie pour identifier l'opinion des intervenants dans une conversation [Hurwitz and Mallery, 1995] en 1995

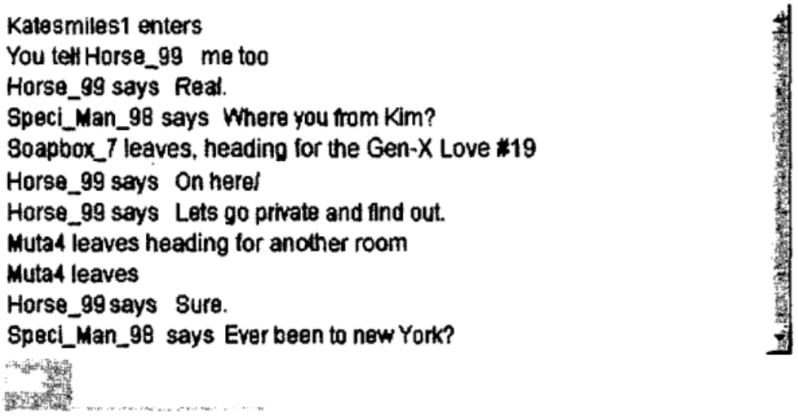


FIGURE 3.10 – Logs dans une messagerie instantanée de groupe pour représenter l'information *Presence History* en 1999 [Viégas and Donath, 1999]

un navigateur intitulé *The Curious Browser*. A partir de ce navigateur, ils calculent le temps passé par les utilisateurs dans les déplacements des curseurs de leurs souris, le nombre de clics effectués, ainsi que les scroll effectués sur la page, ou encore le nombre de saisies clavier. Même si les auteurs ne ciblent pas directement l'awareness, leurs travaux pourraient en faire partie puisque ces informations s'avèrent intéressantes pour calculer le niveau d'implication d'un utilisateur dans un travail collaboratif.

A partir de différents travaux datant des années 70 sur l'eye-tracking montrant que les mouvements des yeux et la fixation sont en rapport di-

rect avec l'intérêt des personnes[Just and Carpenter, 1976] [Kahneman, 1973], d'autres chercheurs sont capables de déterminer précisément les différentes endroits d'un écran captant l'attention des personnes [Bolt, 1990].

Emotional feelings

L'affichage des émotions est également très répandue dans les outils de discussions instantanées. Les émotions sont un facteur important dans la communication [Tran et al., 2005]. La représentation des émotions se présente généralement sous plusieurs formes. La figure 3.11 montre une première représentation sur laquelle des émoticones sont ajoutée à coté du nom de l'utilisateur lorsqu'il souhaite partager son humeur avec ses contacts. Les émotions peuvent également être prises en considération lorsque les personnes utilisent des smileys affectant l'importance accordée aux messages[Rivera et al., 1996].

Availability

Une nouvelle fois, la fonctionnalité de disponibilité se trouve sur les messageries instantanées. Les utilisateurs peuvent indiquer aux autres utilisateurs ce qu'ils sont en train de faire[Tran et al., 2005], par exemple *au téléphone*, *parti* et indiquent donc leur disponibilité à être dérangé. Viegas et Donath affichent cette information au travers de cercles de couleur et de tailles différentes, indiquant le niveau d'activité des utilisateurs. Plus le cercle est gros, plus son activité est intensive et récente.

Roles and responsibilities

La gestion de rôles et de responsabilités implique tout d'abord la notion de droits d'accès et de privilèges. Selon les droits attribués aux utilisateurs, ceux-ci n'auront pas accès aux mêmes fonctionnalités. EVE (Educational Virtual Environment)[Bouras et al., 2003] met en avant ce type de système au travers d'une communauté pour la gestion de cours pour les étudiants. Plusieurs rôles ont été définis : visiteur, membre, étudiant, tuteur, responsable de cours et EVE administrateur. Chacun de ces rôles permet d'effectuer des actions différentes. Celles-ci sont les suivantes :

FIGURE 3.11 – Représentation de l'information *Emotional feelings* dans l'outil de discussion instantané proposé par [Tran et al., 2005]

Place	Rights	EVE Administrator	Course manager	Tutor	Student	Member
Organization	Create Organization	v				
	Delete Organization	v				
	Edit Organization's info	v	v			
	Assign course managers	v				
Course category/ subcategory	Create Course Categories		v			
	Delete Course Categories		v			
	Edit Course Categories info		v			
	Create Course Subcategories		v			
	Delete Course Subcategories		v			
	Edit Course Subcategories info		v			
Courses	Create Courses		v	v		
	Delete Courses		v			
	Assign Tutors		v			
	Validate Course creation		v			
	Edit Courses' info		v	v		
	Accept / Delete Students		v			
	Register for Course			v	v	v
	Attend Course			v	v	

Bilan

Certaines solutions proposées ne sont actuellement pas réalisables. A l'heure actuelle, les techniques d'eye-tracking ne sont pas encore largement déployées sur les ordinateurs. Le problème majeur actuel de l'eye-tracking est la nécessité d'une phase d'apprentissage et de calibrage importants[Zhang et al., 2015]. Néanmoins, la tendance est à la diffusion sur les smartphones. Par exemple, le Samsung Galaxy S4 proposait de l'eye-tracking pour couper l'écran lorsque la personne ne le regarde plus. Apple quant à lui a déposé un brevet en 2015[Julian, 2015] dans le but de manipuler les IHM par les clignements des yeux en ajoutant l'eye-tracking sur ses écrans.

De plus cette solution ou encore celle d'envoyer le positionnement du curseur de souris à chaque modification impliquerait une forte sollicitation du réseau pour échanger ces informations, notamment lorsque des centaines d'utilisateurs sont connectés en même temps à l'outil collaboratif.

Les propositions apportées ne couvrent pas tous les éléments d'awareness. En effet, les aspects sociaux ne sont pas supportés tout comme les informations relatives au passé.

Avant de trouver une solution pour l'affichage de chaque information d'awareness, il est primordial de déterminer les plus importantes pour les utilisateurs. En effet, l'affichage de toutes les informations n'est pas possible sous peine de surcharge cognitive, notamment lorsque toutes les informations arrivent à l'utilisateur automatiquement[Kirsh, 2000]. Il est donc important de n'afficher que les informations essentielles ce qui renforce l'intérêt de la Q3 présentée en introduction de la thèse. Les différentes observations viennent aussi spécifier la Q4 qui devient :

Q4
Comment représenter visuellement les informations d'awareness ? Est-il possible d'appliquer les propositions générales ou de réutiliser les fonctionnalités collaboratives d'autres domaines pour afficher les informations d'awareness ?

Une autre question est également soulevée :

Q5
Les informations sociales et relatives au passé sont-elles importantes ?

Après avoir vu les grands principes de la collaboration et sa composante principale, l'awareness, ainsi que des premières réponses au travers de fonctionnalités, voyons maintenant comment est supporté l'awareness dans les outils et les travaux de recherche.

4 Awareness dans la collaboration synchrone

Lors du chapitre précédent, nous avons pu voir les fonctionnalités collaboratives permettant un support des informations d'awareness. Ce chapitre présente le support de l'awareness dans les outils de modélisation commerciaux et les travaux de recherche. Les informations d'awareness affichées sont différentes selon si la collaboration est synchrone ou asynchrone [Kirsch-Pinheiro et al., 2003]. L'état de l'art du support de l'awareness se portera donc uniquement sur les travaux et outils sur la collaboration synchrone. Cette étude permettra de voir quelles sont les fonctionnalités collaboratives généralement admises pour les outils de modélisation. Nous pourrons ainsi voir quelles sont les limites et les informations d'awareness affichées pour chaque fonctionnalité. Cacoo [Nulab, 2015], Creately[Cinergix, 2015] et LucidChart[Software, 2015] sont des outils de dessin en ligne. Ils sont considérés comme outil de dessin puisque les dessins définis au travers de ces outils ne sont pas valides au sens UML comme j'ai pu le montrer dans la première partie de cette thèse avec la figure 1.5. SLIM[Thum et al., 2009] et GEMSjax [Farwick et al., 2010] ont été présentés dans la première partie de la thèse. Pour rappel, SLIM est un outil web pour l'édition de modèles collaborative de manière synchrone. GEMSjax quant à lui, est un outil pour l'édition de méta-modèles. Nous étendrons notre étude aux fonctionnalités rencontrées dans le développement logiciel et dans l'outil Google Drawings. Nous allons maintenant présenter les fonctionnalités collaboratives communes ou non de ces outils et travaux et voir les éléments d'awareness qu'ils couvrent.

4.1 Présentation des fonctionnalités collaboratives

Focus

Lorsqu'un utilisateur sélectionne un élément de modèle, il est mis en valeur auprès des autres utilisateurs pour indiquer la prise de focus. La couleur de ce focus est différente pour chaque utilisateur. La figure 4.1 illustre le focus dans différents outils.

Limites :

Le focus n'est visible des autres collaborateurs que s'ils travaillent sur des parties de modèles proches. De plus, lors d'une prise de focus, il n'est pas possible de savoir dans quel but il est pris. Par exemple, si un

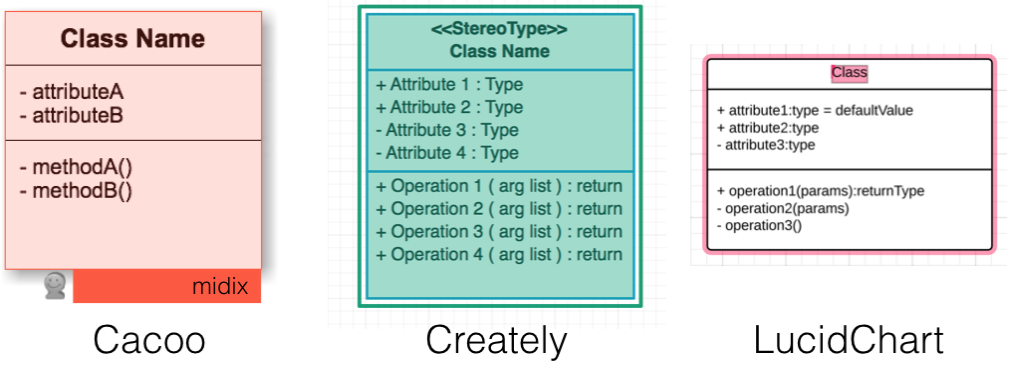


FIGURE 4.1 – Représentation du focus dans les outils de modélisation synchrones (en rouge pour Cacoo, vert pour Creately et rose pour LucidChart)

collaborateur sélectionne une classe, on ne peut pas savoir si c'est dans le but de la supprimer, d'ajouter un attribut...

Eléments d'awareness supportés :

Authorship, Action(partiel), Intention(partiel), Artifact(partiel), Location(partiel)

Travaux et outils proposant la fonctionnalité

Cacoo, Creately, LucidChart, GEMSjax, SLIM

Fenêtre de discussion

La fenêtre de discussion permet de visualiser les collaborateurs actuellement connectés au projet comme l'illustre la figure 4.2. Une couleur est également attribuée à chaque collaborateur, cette couleur correspondant à celle du focus.

Limites :

Aucune

Eléments d'awareness supportés :

Identity, Authorship, Presence

FIGURE 4.2 – Représentation du chat dans les outils de modélisation synchrones

Travaux et outils proposant la fonctionnalité

Cacoo, Creately, LucidChart, SLIM

Visualisation en temps-réel des changements

Les modifications sur le modèle apparaissent immédiatement aux autres collaborateurs. Ils peuvent ainsi suivre les créations, les suppressions, les changements...

Limites :

Les collaborateurs peuvent suivre les actions mais uniquement s'ils travaillent dans la même zone que les autres collaborateurs. Pour que le support de l'awareness soit complet, il faudrait qu'une vue générale représentant les changements ayant lieu sur tous les diagrammes soit ajouté.

Eléments d'awareness supportés :

Action(partiel)

Travaux et outils proposant la fonctionnalité

Cacoo, Creately, LucidChart, GEMSjax, SLIM

Droits

Lorsqu'un utilisateur, propriétaire d'un modèle, le partage avec ses collaborateurs, il peut définir des droits comme le droit de visualiser le modèle uniquement, de le modifier ou de commenter.

Limites :

Les droits sont limités (visualisation uniquement, modification ou commentaires) ce qui implique que l'utilisateur ne peut pas réellement anticiper les actions des autres collaborateurs si ceux-ci ont les droits de modification.

Eléments d'awareness supportés :

Reach(partiel)

Travaux et outils supportant la fonctionnalité

Cacoo, Creately, LucidChart

Historique

Que ce soit Cacoo ou LucidChart, l'historique n'affiche que les versions majeures du modèle. Ils effectuent des sauvegardes régulières et ce sont ces sauvegardes qui apparaissent dans l'historique comme l'illustre la figure 4.3. Cacoo propose un historique textuel. Les dernières sauvegardes apparaissent en indiquant quand elles ont eu lieu et par qui elles ont été effectuées. LucidChart propose en plus de l'historique textuel une interaction permettant de voir l'état du dessin au moment de la sauvegarde.

Limites :

En ne présentant que les versions majeures, une perte d'information est provoquée. De plus, lorsque plusieurs personnes travaillent sur une même version, il n'est pas possible de différencier les actions réalisées par chacun.

Eléments d'awareness supportés :

Action History(partiel), Artifact History(partiel)

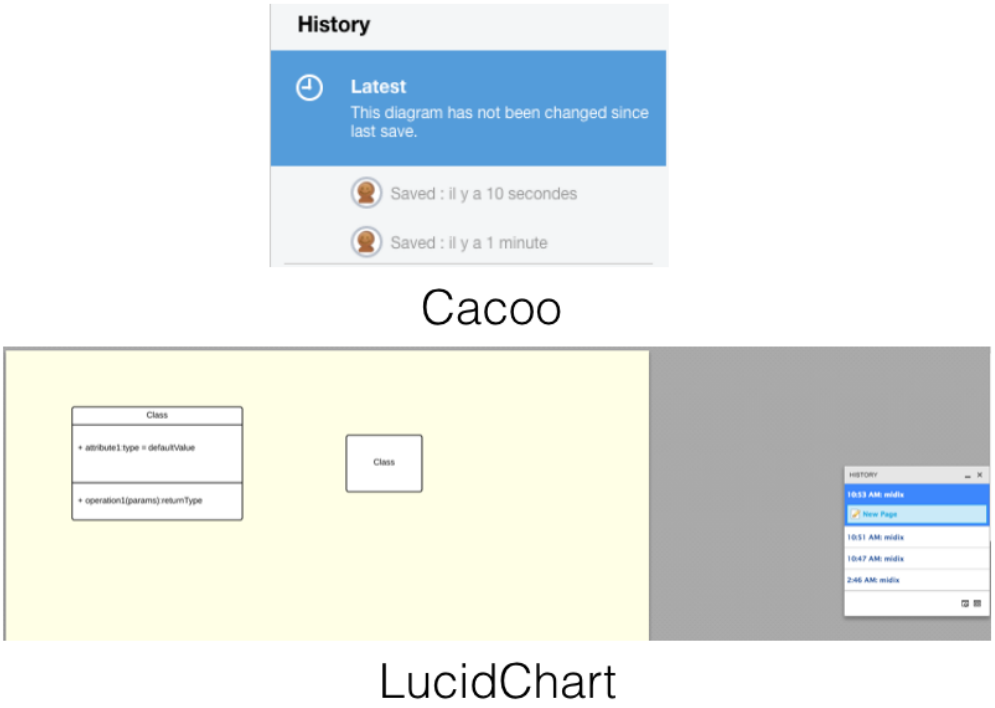


FIGURE 4.3 – Représentation de l'historique dans les outils de modélisation synchrones

Travaux et outils proposant la fonctionnalité

Cacoo, LucidChart

Commentaires

Cette fonctionnalité permet de laisser des commentaires à d'autres collaborateurs comme le montre la figure 4.4. En fonction de l'utilisation de la fonctionnalité, il est possible de prévenir les collaborateurs des actions qui vont être réalisées ou demander l'avis.

Limites :

Ce n'est qu'en fonction de l'utilisation que certaines informations d'awareness vont être données.

Eléments d'awareness supportés :

Intention(partiel), Intention History(partiel), Opinion

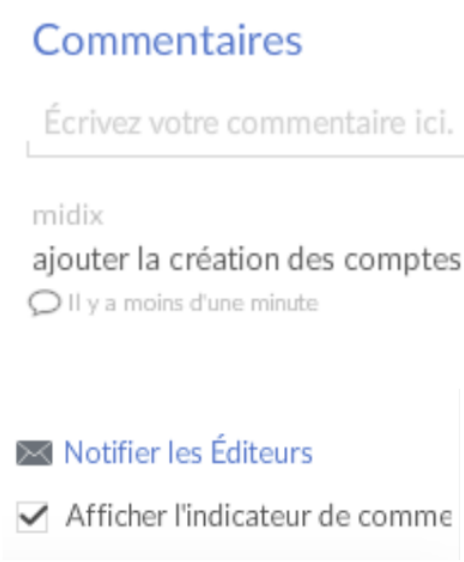


FIGURE 4.4 – Représentation des commentaires dans les outils de modélisation synchrones

Travaux et outils proposant la fonctionnalité

Cacoo

Résumé du support de l'awareness dans les outils de modélisation synchrones

Comme le montre le tableau 4.1, très peu d'informations d'awareness sont complètement supportées. Le support des informations d'awareness n'étant pas complet au sein des outils de dessin et des travaux sur la modélisation, j'ai observé le support de l'awareness dans le développement logiciel, étape suivant la modélisation dans le processus de réalisation d'un logiciel.

4.2 Développement logiciel

La majorité des travaux de recherche sur le support de la collaboration dans le développement logiciel se portent sur la coordination. Pour rappel, la coordination concerne l'organisation et la gestion des tâches ainsi que des collaborateurs. Pour se coordonner, les membre des équipes doivent connaitre les tâches à réaliser et leur interdépendances

TABLE 4.1 – Support de l'awareness dans les outils de modélisation synchrones

	Outil				
	Cacoo	Creately	LucidChart	GEMSjax	SLIM
Identity	✓	✓	✓		✓
Authorship	✓	✓	✓		✓
Action	❍	❍	❍	❍	❍
Action history	❍				
Intention	❍	❍	❍	❍	❍
Intention history		❍			
Artifact	❍	❍	❍	❍	❍
Artifact history	❍		❍		
Location	❍	❍	❍	❍	❍
Location history	❍				
Gaze					
View					
Reach	❍	❍	❍		
Event history					
Opinion	✓				
Presence	✓	✓	✓		❍
Presence history					
Interest level					
Emotional feelings					
Availability					
Roles and responsibilities					

✓ : information fournie
❍ : information fournie partiellement

[Damian et al., 2007]. Pour cette raison, de nombreux articles se portent sur la gestion de ces tâches. TeamSpace[Geyer et al., 2001] adresse ces problématiques en proposant aux utilisateurs la gestion des personnes, des tâches et du projet. L'outil permet également de planifier des meetings et d'enregistrer des documents relatifs à ce meeting comme des enregistrements vidéos, des PowerPoint annotés. TeamSpace propose une interface montrant les personnes connectées ainsi que leur disponibilité et prend une capture de l'écran des utilisateurs toutes les minutes. Ainsi, les informations d'awareness affichées par les fonctionnalités sont les suivantes :

Presence : Panneau des utilisateurs.

Availability : Panneau des utilisateurs.

Location : Captures d'écran.

Intention : Captures d'écran et gestion des tâches.

Gutwin propose ProjectWatcher[Gutwin et al., 2005], un plug-in Eclipse traçant les changements réalisés par les utilisateurs au travers d'un outil de gestion de version. ProjectWatcher est issu des observations de Gutwin concernant le besoin d'awareness dans plusieurs projets distribués. Il a trouvé que les développeurs maintiennent une connaissance générale de *Qui est qui* et de *Qui travaille sur quoi dans le projet*. Ils suivent également les changements réalisés par d'autres développeurs lorsque ceux-ci travaillent de manière proche à leur travail. Avec les in-

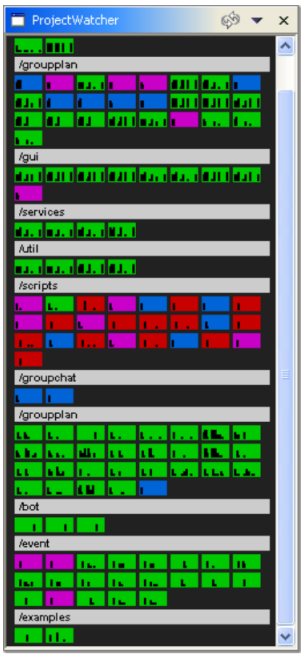


FIGURE 4.5 – Affichage des informations relatives au Workspace Awareness dans ProjectWatcher

formations contenues dans l'outil de gestion de version et des actions en temps-réel dans Eclipse, ProjectWatcher est capable d'afficher des informations comme l'activité des utilisateurs, les artéfacts qu'ils sont en train d'utiliser et où ils ont travaillé dans le projet comme le montre la figure 4.5. Le widget présenté montre les différents packages en gris et des rectangles colorés correspondant aux classes avec comme couleur la

dernière personnes ayant modifié une classe. Project Watch adresse donc les informations d'awareness suivantes :

Toutes les informations relatives au Workspace Awareness hormis Gaze, View et Reach

Interest Level

D'autres travaux se basent sur le dépôt de code pour fournir des informations aux équipes de développement. Tel est le cas de Manhattan [Lanza et al., 2013], un plug-in Eclipse également, fournissant la visualisation des informations en temps-réel suivantes :

— conflits potentiels dans lesquels un développeur est impliqué
— les classes ayant été changées par d'autres développeurs
— les développeurs modifiant une classe de manière fréquente.

La représentation de ces informations est originale puisque le projet est représenté par une ville et les classes par des bâtiments. Les bâtiments étant colorés en fonction des conflits possibles et des actions réalisées par les autres développeurs comme le montre la figure 4.6.

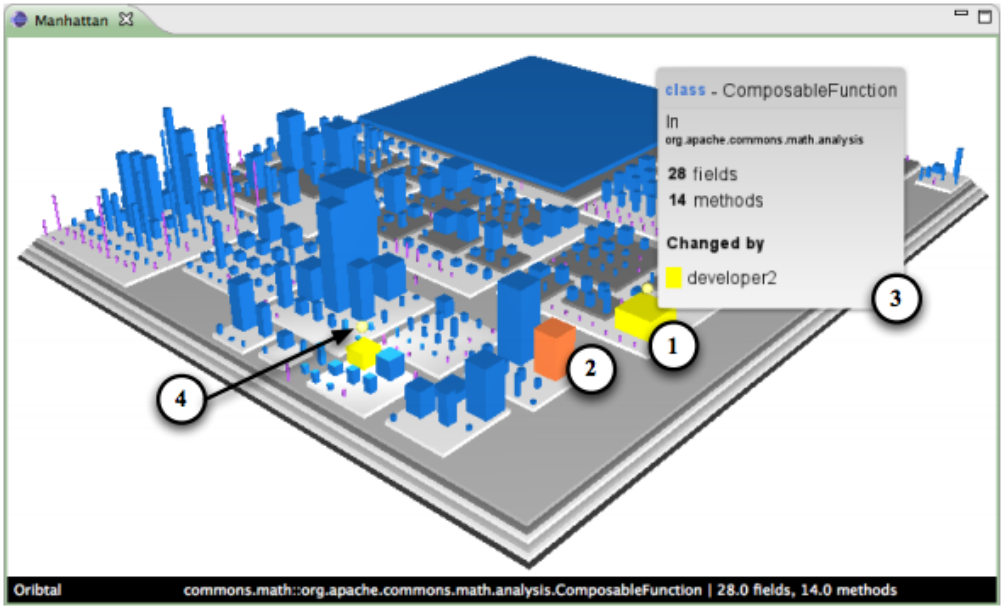


FIGURE 4.6 – Représentation des conflits possibles et des actions réalisées par les autres développeurs dans Manhattan

Manhattan se base sur Syde[Hattori, 2010][Hattori and Lanza, 2009a] [Hattori and Lanza, 2009b][Hattori and Lanza, 2010][Hattori et al., 2012], seule la visualisation des informations est différente.

Le même type de représentation est utilisé dans SecondWATCH [Ye et al., 2009]. Cette fois, les "bâtiments" sont représentés par des cylindres empilés comme le montre la figure 4.7. Lorsque des changements

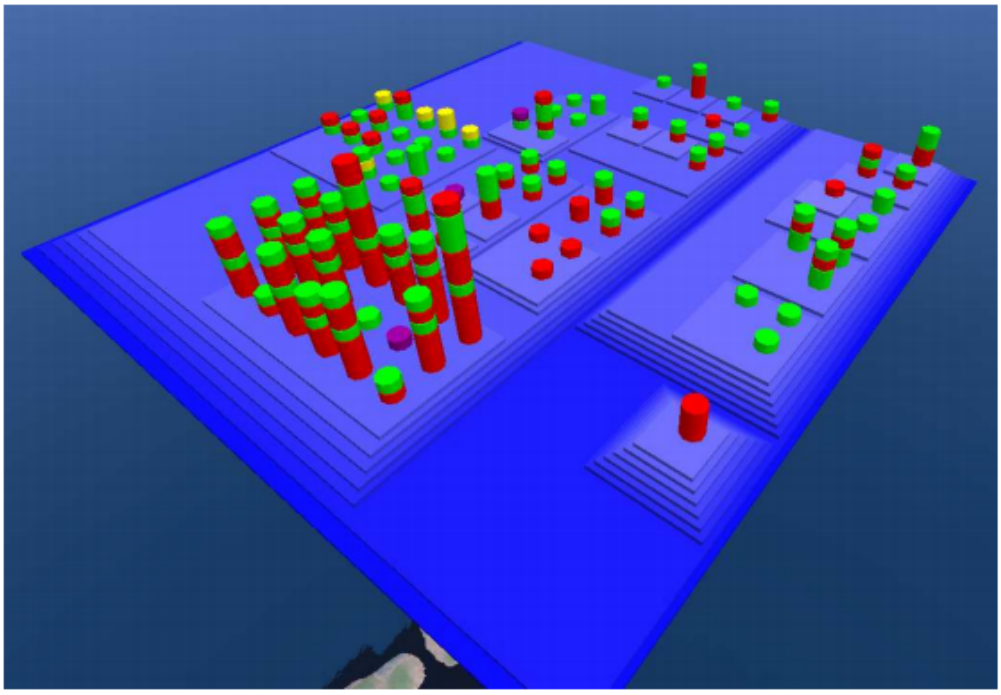


FIGURE 4.7 – Activité sur un projet dans SecondWATCH

sont en cours sur une classe, de la fumée est rajoutée au dessus du cylindre. Les solutions Manhattan et SecondWATCH permettent d'adresser les informations d'awareness **relatives au Workspace Awareness**(hormis Gaze, View et Reach) et **Interest Level**.

FASTDash[Biehl et al., 2007] propose un dashboard permettant de suivre l'activité des utilisateurs sur le dépôt de code comme le montre la figure 4.8. Ce dashboard utilise différentes variables visuelles (bordures, textures, couleurs) ainsi que la photo des utilisateurs pour montrer les fichiers en cours d'édition, ceux qui ont été récupérés, ceux ouverts par d'autres utilisateurs. FASTDash permet donc d'afficher les informations :

Action : Rectangle jaune.

Intention : Rectangle jaune.

Artifact : Nom de la classe.

Identity : Photo du développeur.

Authorship : Photo du développeur

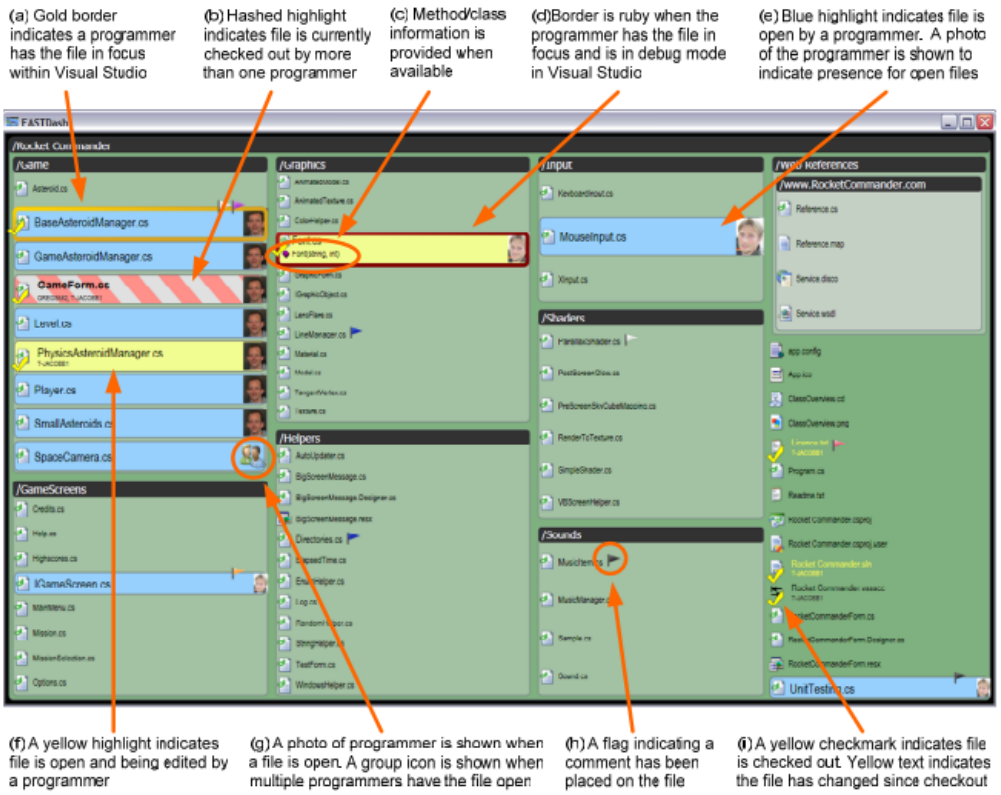


FIGURE 4.8 – Représentation de l'activité des utilisateurs dans FASTDash

4.3 Google Drawings

Une nouvelle fois, je vais présenter les fonctionnalités collaboratives d'un des outils collaboratifs les plus utilisés par le géant du web américain Google et notamment Google Drawings comme nous avons déjà pu le voir en première partie de cette thèse. Google Drawings ne se positionne pas sur le marché des outils UML, contrairement à Cacoo, LucidCart ou Creately.

La figure 4.9 montre les différentes fonctionnalités. La collaboration démarre par la fenêtre de partage (1) dans laquelle l'utilisateur saisit le nom du collaborateur ou son adresse mail et lui attribue des droits : droits de modification, de commentaire ou de lecture.

Lorsque ce collaborateur rejoindra le document, une icône apparaitra avec la photo de profil de celui-ci (2) pour notifier de sa présence. Sous cette image un rectangle de couleur est présent et correspond à la couleur qui lui est attribuée. Cette couleur est utilisée lorsque le collaborateur

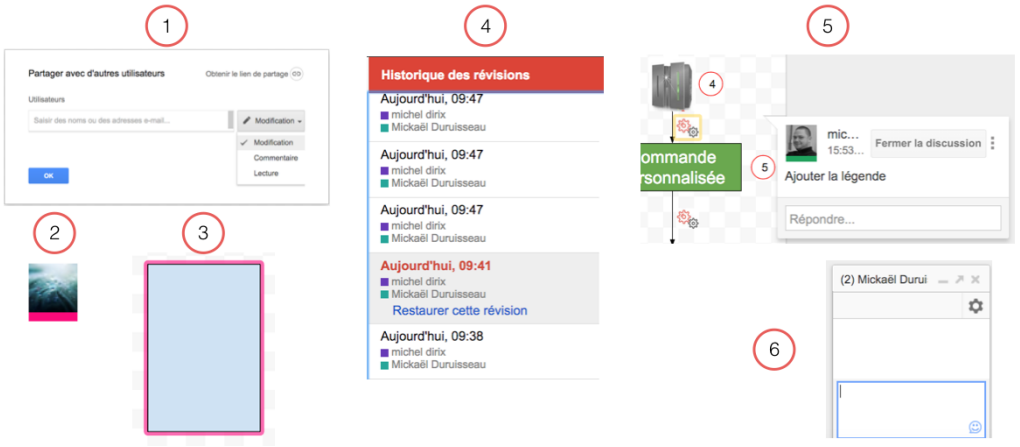
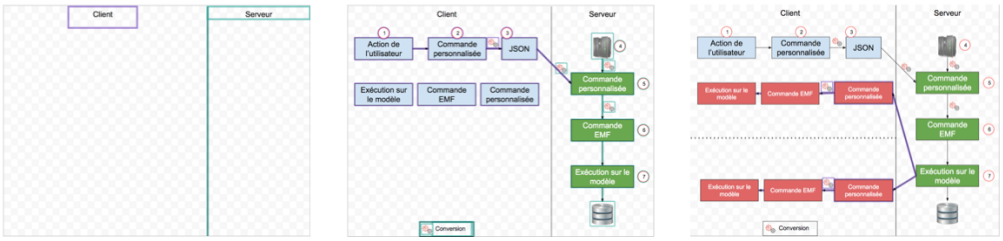


FIGURE 4.9 – Fonctionnalités collaboratives dans Google Drawings

sélectionne des éléments (3).

Les collaborateurs peuvent ensuite lancer un historique des révisions dans le but de voir les modifications apportées au document dans le temps. La figure 4.10 montre la vue du schéma lors de la visualisation des différentes versions. Le passage de la version 10 à la version 11 s'avère problématique. En effet, ce passage ne se fait pas avec une granularité fine puisqu'un bloc de modifications apparait en version 11. Cette solution peut être adaptée à la définition de schémas de ce type mais la granularité n'est pas assez fine pour un outil de modélisation. La séquence logique de construction n'apparait pas alors qu'en affichant l'intégralité des actions, il serait possible de déterminer cette logique et de comprendre le but des actions réalisées. D'autant plus, qu'un projet de modélisation ne se résume pas à un seul document, la complexité de compréhension est encore plus grande pour la modélisation.



Version 10 Version 11 Version finale

FIGURE 4.10 – Détail de l'utilisation de l'historique Google Drawings

Ensuite, les collaborateurs ont la possibilité de commenter le schéma (5). Chaque collaborateur (ayant les droits) peut ainsi répondre aux commentaires et ensuite les clôturer.

Enfin, les collaborateurs peuvent discuter grâce à la fenêtre de chat (6).

Ces différentes fonctionnalités permettent de supporter un grand nombre d'informations d'awareness :

Identity : Liste des utilisateurs connectés.

Presence : Liste des utilisateurs connectés.

Authorship : Liste des utilisateurs connectés et focus.

Action : Changements en temps réel.

Action History (partiel) : Historique.

Artifact : Changements en temps réel.

Artifact History (partiel) : Historique.

Intention (partiel) : Focus + commentaires.

Intention History (partiel) : Commentaires.

Location : Focus.

Location History (partiel) : Historique.

View : Focus.

Opinion : Chat et commentaires.

Google Drawings affiche plus d'informations d'awareness sans pour autant fournir plus de fonctionnalités collaboratives que les outils de modélisation. Ceci vient du fait que sur Google, les utilisateurs n'ont une vue que sur un seul dessin, contrairement à la modélisation qui s'étend sur plusieurs documents. Une nouvelle fois, la modélisation s'avère plus complexe et la solution adoptée dans Drawings doit donc être adaptée.

4.4 Discussions

Les travaux présentés dans ce chapitre ont confirmé les dires d'Omoronyia [Omoronyia et al., 2010] : le support de l'awareness est principalement centré sur le *Workspace Awareness*. Le tableau 4.2 montre les informations d'awareness affichées pour chacun des travaux présentés. Dans les outils de modélisation, le support des informations, même pour le *Workspace Awareness* n'est pas suffisant. En effet, la plupart des informations ne sont pas intégralement prises en compte, notamment pour les informations se référant au passé. Sur ce point, les travaux concernant

le développement proposent un meilleur support en se centrant sur les tâches et sur les modifications en cours de réalisation grâce au système de cylindres montrant l'intensité de l'activité sur les classes. Ce système doit néanmoins être couplé au dépôt de code pour obtenir l'intégralité des informations, ce qui empêche son intégration en l'état pour les outils de modélisation. Google Drawings propose un support équivalent aux outils de modélisation. Le système de focus seul ne permet pas une gestion efficace des informations *Action* et *Artifact*. De plus, la gestion de l'historique ne montre pas toutes les modifications ayant eu lieu et gène la compréhension de la logique de construction.

L'analyse des travaux précédents permet de mettre également en évidence un point : le support de toutes les informations d'awareness implique l'affichage d'une multitude de widgets. Lorsqu'une personne est amenée à traiter trop d'informations, elle est victime de ce que l'on appelle un *Cognitive Overload*[Mayer and Moreno, 2003]. Plusieurs raisons pouvant provoquer cette saturation ont été montrées[Kirsh, 2000] :

— Trop d'informations en *pull* (la personne va chercher les informations au serveur).

— Trop d'informations en *push* (le serveur envoie les informations à la personne).

— Le multi-tâches.

— Les interruptions.

— Un environnement de travail mal conçu.

Pour éviter l'apparition de ce problème, il est donc nécessaire de limiter le nombre d'informations à afficher aux utilisateurs mais aussi de les afficher au bon moment pour ne pas le déranger dans son activité. Pour ce faire, il faut donc déterminer quelles sont les informations importantes à lui afficher et dans quels contextes elles doivent l'être. Cette étude préliminaire n'a jamais été réalisée, j'ai donc décidé de la réaliser pour affiner la compréhension du problème de la modélisation collaborative.

TABLE 4.2 – Résumé sur le support de l'awareness dans les outils de modélisation, de développement et Drawings

	Modélisation						Développement					Autre
	Cacoo	Creately	LucidChart	GEMSjax	SLIM	TeamSpace	Project Watcher	Manhattan	SecondWATCH	FASTDash	Drawings
Identity	✓	✓	✓		✓		✓	✓	✓	✓	✓
Authorship	✓	✓	✓		✓		✓	✓	✓	✓	✓
Action	○	○	○	○	○		✓	✓	✓	✓	✓
Action history	○										
Intention	○	○	○	○	○		✓	✓	✓	✓	○
Intention history		○					✓	✓	✓		○
Artifact	○	○	○	○	○		✓	✓	✓	✓	○
Artifact history	○		○				✓	✓	✓		○
Location	○	○	○	○	○	✓	✓	✓	✓		○
Location history	○						✓	✓	✓		○
Gaze											✓
View											
Reach	○	○	○								
Event history											✓
Opinion	✓										✓
Presence	✓	✓	✓		○						
Presence history							✓	✓	✓		
Interest level											
Emotional feelings											
Availability						✓					
Roles and responsibilities											

✓ : information fournie
○ : information fournie partiellement

5 Détermination des types de collaboration

La première partie de cette thèse a permis de mettre en avant comment les utilisateurs peuvent se communiquer les changements dans le but de collaborer. Pour assister les utilisateurs dans la collaboration, il est nécessaire de comprendre comment ils travaillent pour fournir une solution proche de leurs besoins. Dans le cas de l'édition de documents collaborative, Posner et Baecker[Posner and Baecker, 1992] ont réalisé ce travail et ont déterminé les différents types de collaborations utilisés et sont au nombre de quatre :

Single Writer : Une personne est en charge de l'écriture en se basant sur les discussions avec les autres membres du groupe.

Scribe : Ce mode est utilisé lors de réunions où l'une des personnes se charge d'écrire les idées du groupe.

Separate Writers : Le document est divisé en différentes parties. Chaque personne est en charge de l'une d'entre elles.

Joint Writing : Les membres du groupe écrivent le document ensemble, décidant ensemble du contenu.

Ce genre d'étude n'a pas été réalisé pour la modélisation collaborative. La première partie de mon travail pour assister les utilisateurs dans cette collaboration a donc été de déterminer comment les utilisateurs collaborent, pour ensuite pouvoir déterminer les informations d'awareness à afficher pour chacun de ces modes. Comme nous avons pu le voir précédemment, il n'est pas possible d'afficher toutes les informations d'awareness sous peine de provoquer un *Cognitive Overload* de l'utilisateur. La détermination des différents modes de collaboration permettra de juger les informations d'awareness impliquées dans chacun de ces modes.

Après la mise en place de la collaboration au sein de l'outil GenMyModel, j'ai pu analyser les sessions des utilisateurs. En effet, pour permettre l'échange des modifications entre les utilisateurs, nous avons mis en place le système dAMOCleS, définissant des commandes pour représenter ces changements. De plus, lorsqu'un utilisateur rejoint son projet, un événement de connexion est enregistré en base de données. De ce fait, il est possible de reconstruire la session d'activité d'un utilisateur à partir de cet événement et des changements réalisés.

5.1 Démarche

Afin de définir les différents types de collaboration, j'ai sélectionné des projets collaboratifs selon certains critères :

— Les projets devaient être publics pour pouvoir en analyser le contenu.

— Ils ne devaient pas être clonés à partir d'un projet d'exemple pour éviter les projets "tests" des utilisateurs. Les projets d'exemples servant de point de départ aux utilisateurs pour découvrir rapidement les fonctionnalités de GenMyModel.

— Ils devaient être récents pour pouvoir interroger les utilisateurs sur un projet dont ils avaient encore souvenance.

Selon ces critères, 508 projets collaboratifs ont ainsi pu être analysés. Pour chaque projet, j'ai reconstruit les sessions d'activités en fonction des commandes qui ont été lancées. Pour analyser ces sessions, un critère a dû être défini :

Une inactivité de plus d'une minute entraine la fin de la session .

Un programme a ensuite été développé pour reconstruire les sessions de manière visuelle en utilisant Google Chart [1]. Une fois les sessions reconstruites, l'analyse s'est faite de manière manuelle pour déterminer les différents modes de collaboration, présentés maintenant. Voici les résultats.

5.2 Un seul contributeur (OC)

Ce type de collaboration implique plusieurs collaborateurs mais un seul effectue des modifications sur le modèle. Celui-ci a été nommé One Contributor(OC). La figure 5.1 montre un exemple de ce type de collaboration dans lequel Michel a effectué 939 modifications sur le modèle sur un intervalle de temps de 2h. Ce type de collaboration est le plus

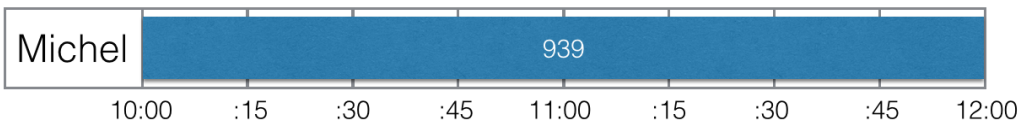


Figure 5.1 – Exemple de session avec un seul contributeur

souvent employé. En effet, sur les 508 projets satisfaisant les critères,

1. https ://developers.google.com/chart/interactive/docs/gallery/timeline

308 sont dans cette configuration (61%). Une seule personne effectue les modifications, néanmoins, les autres membres de l'équipe se connectent régulièrement pour voir l'état du modèle.

5.3 Plusieurs contributeurs

Lorsque plusieurs contributeurs agissent sur le modèle. Ils alternent entre deux modes de collaborations. Les projets regroupant plusieurs contributeurs représentent 39% des cas étudiés. Dans le premier mode, les collaborateurs se relaient pour effectuer les modifications sur le modèle. Ce type de collaboration a été appelé Turn By Turn (TBT). La figure 5.2 montre un exemple dans lequel Michel commence par effectuer 20 modifications sur le modèle puis stoppe son activité. Xavier prend ensuite le relai pour réaliser 42 modifications et ainsi de suite. L'intervalle entre deux changements d'auteur peut être de quelques secondes, minutes ou heures. Lorsque le délai est très court, le critère défini auparavant sur le temps d'inactivité d'un utilisateur permet de classer dans l'un des deux modes, selon les règles suivantes :

— Le temps écoulé entre la modification d'un auteur *a1* et celle d'un auteur *a2* est supérieur à une minute : classé dans le mode TBT.

— Sinon la collaboration est classée dans le second mode.

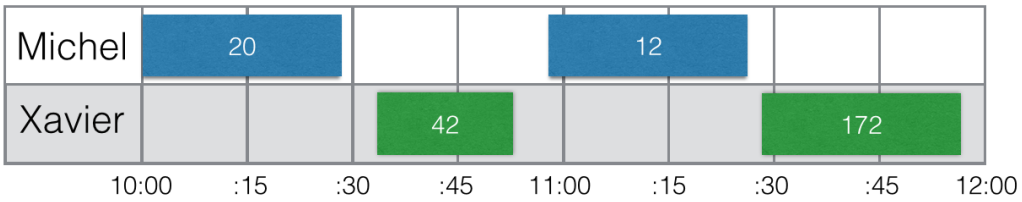


Figure 5.2 – Exemple de session en tour par tour

Dans ce second mode, plusieurs membres de l'équipe réalisent des modifications sur le modèle en même temps. Ce type de collaboration a été appelé Real-Time (RT). La figure 5.3 montre que Michel et Xavier ont modifié le modèle en même temps. Ce type est très peu utilisé. Le RT ne représente que 5% du temps dans ces 39% de projets collaboratifs. Ce temps peut être crucial pour la coordination entre les utilisateurs mais il n'y a pas de moyen de le vérifier.

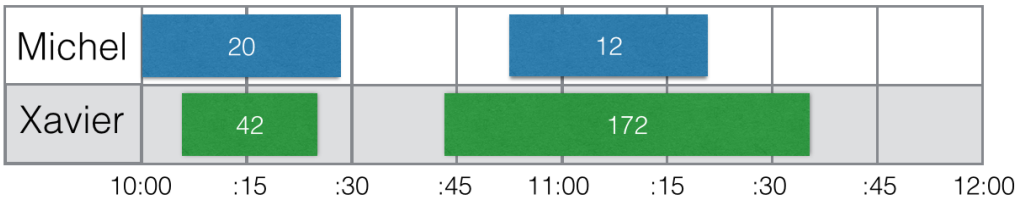


FIGURE 5.3 – Exemple de session en temps réel

Remarque complémentaire

Lors de l'analyse des sessions des utilisateurs, j'ai pu remarquer que le temps entre la connexion de l'utilisateur et sa première action était élevé (plus d'une minute). Plusieurs hypothèses sont à vérifier pour expliquer ce temps élevé :

- Il s'agit du temps nécessaire à l'utilisateur pour se remettre dans le contexte et voir les changements ayant eu lieu sur le modèle depuis sa dernière connexion.
- L'utilisateur fait une autre activité pendant le chargement de son modèle.
- Il s'agit du temps de démarrage de la nouvelle activité de l'utilisateur.

L'étude sur l'activité des utilisateurs a permis de déterminer trois modes de collaboration, nous permettant de commencer à déterminer les informations importantes à afficher. Selon le type de collaboration dans lequel se placent les utilisateurs, il est désormais possible de savoir toutes les informations d'awareness étant impliquées, ce que nous allons présenter maintenant.

5.4 LIAISON AWARENESS/TYPES DE COLLABORATIONS

Après avoir défini les différents types de collaboration possibles en analysant les sessions d'activité des utilisateurs, j'ai "filtré" les éléments d'awareness pour chacun afin de ne garder que ceux étant pertinents.

OC

Ce type de collaboration n'impliquant qu'une seule personne active, le nombre d'éléments d'awareness impliqués est très faible. Seules *l'In-*

tention et *l'Opinion* peuvent être utiles. Les informations relatives aux activités en cours de réalisation et passées n'ont aucun intérêt pour ce mode de collaboration puisqu'il n'y a qu'un seul éditeur. Un cas possible d'utilisation de ce mode de collaboration est le suivant : La personne qui construit le modèle est un architecte et le reste de l'équipe est composée de développeurs. Dans ce cas, les développeurs vont consulter régulièrement le modèle pour en implémenter le code mais aussi pour donner leur avis. Ils vont également vouloir connaitre l'avancement de certaines parties du modèle pour savoir si celle-ci est terminée pour réaliser l'implémentation.

TBT

Pour le TBT, les éléments d'awareness pouvant être le plus utile sont ceux relatifs au passé et ceux concernant les tâches restantes. De ce fait, les informations d'awareness pertinentes sont *Authorship*, *Action History*, *Intention*, *Intention History*, *Artifact History*, *Location History*, *Event History*, *Opinion* et *Presence History*. Lorsque le mode de collaboration est TBT, il n'y a qu'un seul utilisateur actif à la fois. L'utilisateur peut néanmoins avoir besoin des informations passées pour comprendre l'état actuel du modèle. Il n'aura par contre pas besoin des informations *Identity*, *Action*, *Artifact*, *Location*, *Gaze*, *Reach*, *View*, *Presence*, *Interest Level*, *Emotional Feelings*, *Availibility* et *Roles and Responsibilities* puisque toutes ces informations se rapporteraient à lui-même. Il est donc inutile de lui afficher des informations relatives à sa propre activité.

RT

Pour ce type de collaboration, tous les éléments d'awareness peuvent être nécessaires.

En résumé

Le tableau 5.1 résume les éléments d'awareness pertinents pour chaque type de collaboration. Le nombre d'informations à afficher est encore trop élevé, il faut donc déterminer quelles sont les plus importantes pour les différents modes de collaboration et dans quel contexte il est pertinent de les afficher dans le but de fournir la bonne information au bon moment. A partir de la définition des différents types de collaboration et des éléments d'awareness s'y rattachant, j'ai pu réaliser une enquête auprès d'utilisateurs d'outils de modélisation afin de définir les éléments les plus importants. Après avoir déterminé les informations d'awareness

TABLE 5.1 – Elements d'awareness pertinents pour chaque type de collaboration

Element	**OC**	**TBT**	**RT**
Identity			X
Authorship		X	X
Action			X
Action history		X	X
Intention	X	X	X
Intention history		X	X
Artifact			X
Artifact history		X	X
Location			X
Location history		X	X
Gaze			X
View			X
Reach			X
Event history		X	X
Opinion	X	X	X
Presence			X
Presence history		X	X
Interest level			X
Emotional feelings			X
Availability			X
Roles and responsibilities			X

les plus importantes et les différents contextes, nous serons en mesure de proposer des fonctionnalités collaboratives proches des besoins des utilisateurs sans les surcharger d'informations.

6 Contextualisation de l'awareness

Dans le chapitre précédent, nous avons vu qu'il existait trois types de collaborations : un seul contributeur(OC), tour par tour (TBT) et temps-réel (RT). Pour la collaboration RT et TBT, de nombreuses informations sont à afficher, il n'est néanmoins pas possible de toutes les afficher, l'utilisateur risquerait une surcharge cognitive. Pour éviter que ce problème ne survienne, il est nécessaire de n'afficher que les plus importantes au bon moment. Pour ce faire, nous allons classer les informations d'awareness par importance et déterminer différents contextes dans lesquels certaines informations sont plus ou moins utiles. Afin de classer les éléments d'awareness par importance, j'ai réalisé des questionnaires en ligne, diffusés auprès des utilisateurs de GenMyModel et sur des réseaux sociaux auprès d'utilisateurs d'autres outils de modélisation pour généraliser les résultats. Les contextes quant à eux ont été réalisés grâce à une étude statistique. Durant ce chapitre, nous allons commencer par détailler la méthode utilisée pour recueillir les informations importantes, puis je présenterai le contenu des questionnaires mis en place pour ensuite montrer les résultats de l'étude et les différents contextes.

6.1 Protocole

Le but de l'étude est de demander aux utilisateurs de la modélisation de classer les éléments d'awareness par importance les uns par rapport aux autres, après leur avoir expliqué le sens de chacun de manière simple. Ensuite, chaque information est noté de manière atomique pour vérifier la pertinence des réponses données mais aussi pouvoir vérifier les impacts de chaque réponse en fonction du contexte de l'utilisateur. Pour cela, un questionnaire par type de collaboration a été construit (en utilisant Google Forms). Le premier concerne uniquement le type de collaboration OC. Le second concerne les types TBT et RT puisque l'utilisation du TBT et du RT n'est pas exclusive. Pour chacun des deux questionnaires, j'ai posé les questions suivantes :

— Contexte du projet ? Projet étudiant, projet professionnel ou projet de recherche

— Type d'application modélisée ? Application lourde, application web ou application mobile.

— Taille de l'équipe ? 1-3, 3-5, 5-10, 10+.

— A quelle distance vous situez-vous de vos collaborateurs ? Nous partageons le même bâtiment, la même ville, le même pays ou nous sommes dispersés à travers le monde.

Ces questions générales ont été posées pour vérifier si ces critères pouvaient influencer les réponses. Elles seront au centre de l'analyse statistique pour déterminer les différents contextes impactant les besoins d'awareness.

Les questions suivantes sont spécifiques au type de collaboration.

OC

En plus des questions précédentes, j'ai ajouté les questions suivantes :

— Avez-vous besoin de l'opinion de vos collaborateurs ?

— Si oui, dans quel contexte ?

— Comment demandez vous leur avis ?

— Vos collaborateurs savent-ils sur quelle tâche vous travaillez ?

— Si oui, comment le savent-ils ? Comment utilisent-ils cette information ?

La première partie de ces questions sert à déterminer l'importance de l'opinion. Le seconde partie concerne l'intention.

Pour collecter les réponses à ces questions, j'ai diffusé le questionnaire par email auprès des contributeurs des 308 projets que j'avais analysés, faisant partie du type OC.

RT et TBT

Les deux types de collaborations étant utilisés au sein d'un projet impliquant plusieurs contributeurs, j'ai décidé de découper le questionnaire en deux parties. La première partie traite des informations importantes pour le RT et la seconde partie pour le TBT. Pour chaque information d'awareness impliquée pour chacun des deux types de collaboration, je devais évaluer l'importance de l'information. Pour ce faire, j'ai demandé aux utilisateurs de noter l'importance de chaque information d'awareness sur une échelle de 1 à 4 (1 étant inutile et 4 indispensable). J'ai fait le choix d'une échelle paire pour éviter les avis neutres. Pour simplifier le questionnaire et éviter de devoir expliquer chaque information d'awareness, une question plus générale a été posée. Par exemple, au lieu de demander la pertinence de l'information *Identity*, j'ai demandé s'il était pertinent de connaitre le nom des collaborateurs connectés.

En plus de cette échelle pour chaque élément d'awareness, j'ai demandé à l'utilisateur de donner les informations d'awareness qu'il jugeait être les plus importantes. Il devait en citer 5 pour le RT et 3 pour le TBT. Ceci permettant de vérifier qu'il n'y ait pas d'incohérences dans les réponses. Lors de la phase d'analyse des sessions d'activités, j'avais pu remarquer que le délai entre la connexion de l'utilisateur et sa première action était élevé. Une question supplémentaire a donc été ajoutée pour en déterminer la raison.

Pour diffuser ce questionnaire, j'ai d'abord contacté 392 utilisateurs de GenMyModel dont les projets avaient été analysés dans la phase précédente. Lors de cette prise de contact par email, je leur demandais simplement s'ils acceptaient de participer à l'amélioration de l'outil. Après acceptation, je leur envoyais le lien vers le questionnaire. En complément, ce questionnaire a été diffusé sur des réseaux sociaux professionnels concernant la modélisation. En diffusant celui-ci, la cible des personnes n'était plus les utilisateurs de GenMyModel mais des utilisateurs d'outils de modélisation en général. Le but étant de vérifier si l'outil utilisé influençait les résultats.

6.2 Résultats

OC

Pour ce premier questionnaire, 17 personnes ont répondu. La figure 6.1 montre la part de réponses à chaque question.

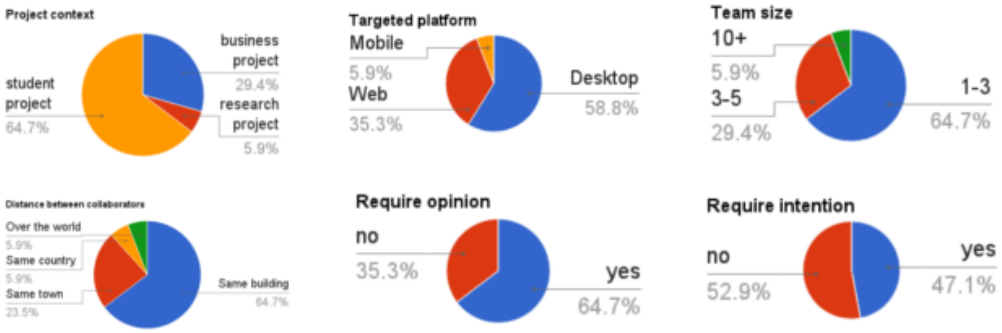


Figure 6.1 – Réponses pour le OC

Les résultats montrent que ce sont les étudiants qui utilisent principalement ce type de collaboration. De plus, la plupart des applications

modélisées sont des applications lourdes. En général, le nombre de collaborateurs est très limité (entre 1 et 3) et les utilisateurs partagent très souvent le même bâtiment. La plupart des contributeurs des projets OC ont besoin de l'avis de leurs collaborateurs dans le but d'améliorer la qualité du modèle. Du point de vue des intentions, les utilisateurs sont plus partagés puisque moins de la moitié partage ses tâches avec les autres membres de l'équipe. Lorsque le partage des tâches est réalisé, il se fait au travers d'outils de gestion de projets, de groupes Facebook ou durant des réunions. Le nombre de personnes ayant répondu au questionnaire étant faible, il n'a pas été possible de faire une analyse statistique plus poussée.

RT et TBT

Pour ce second questionnaire, 50 personnes y ont répondu : 35 sont des utilisateurs de GenMyModel contactés par email, les 15 autres sont des utilisateurs de Papyrus, Enterprise Architect... ayant répondu au questionnaire partagé sur les réseaux sociaux.

Comme le montre la figure 6.2, les informations les plus importantes pour le RT sont la *Presence*, l'*Identity*, l'*Artifact/Action*, l'*Intention* et l'*Authorship*.

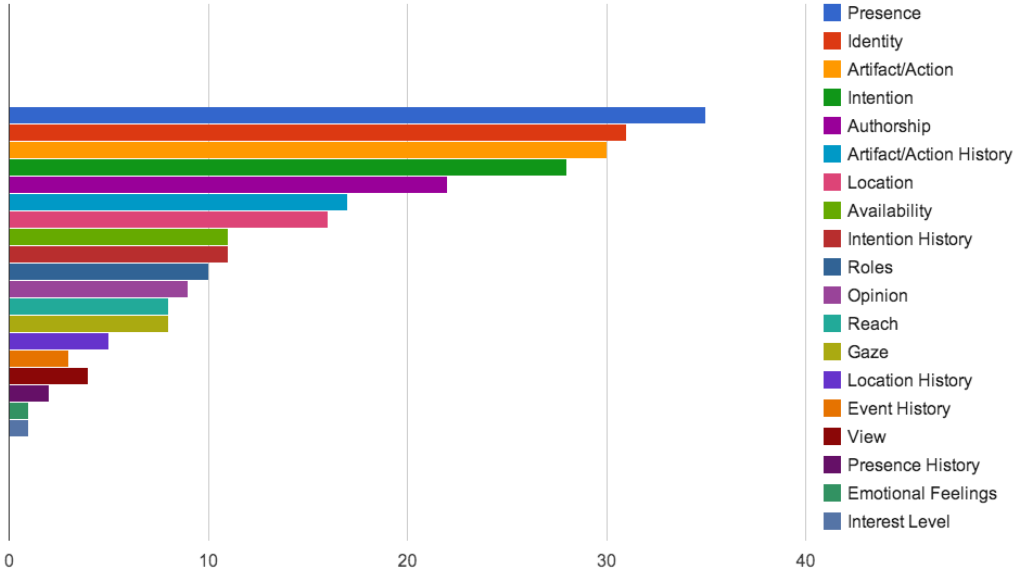


FIGURE 6.2 – Classement des informations d'awareness pour le RT

Les réponses diffèrent concernant le TBT, puisque celles mises en valeur sont l'*Action/Artifact History*, l'*Authorship* et l'*Intention*.

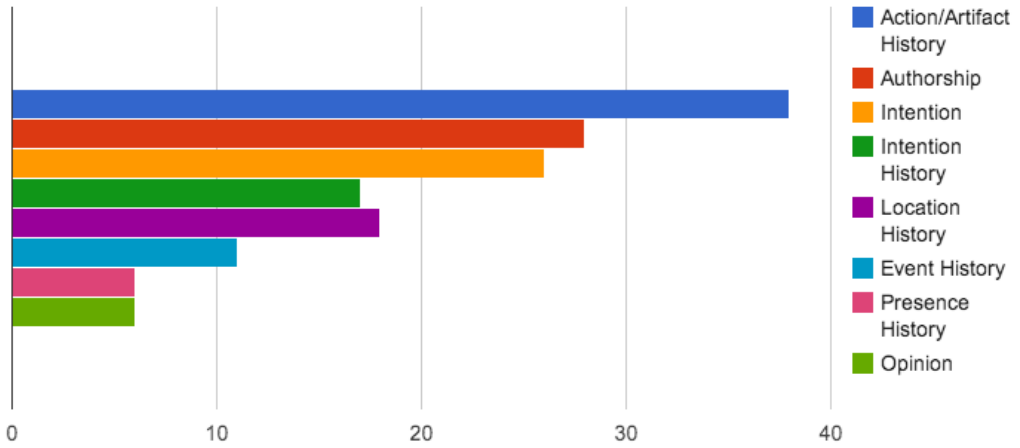


FIGURE 6.3 – Classement des informations d'awareness pour le TBT

6.3 CONTEXTES IMPACTANT LES BESOINS

La première partie du questionnaire portait sur des informations générales telles que la taille de l'équipe, le type d'application modélisée, le contexte du projet, la distance entre les collaborateurs, l'outil de modélisation utilisé. De plus, je connaissais le nombre de commandes exécutées sur chaque projet (pour les utilisateurs de GenMyModel). Toutes ces informations ont permis de réaliser une étude statistique pour vérifier si elles influençaient les résultats. Des tests de corrélation de Pearson ont été effectués entre les informations générales et la valeur attribuée sur l'échelle de 1 à 4 pour chaque élément d'awareness. Les premiers tests de corrélation ont été réalisés entre les variables (taille de l'équipe, taille du projet...) pour vérifier que les variables étaient indépendantes. Les tests ont ensuite été effectués entre chaque variable et chaque élément d'awareness. La figure 6.4 montre les corrélations trouvées pour le RT. La figure 6.5 montre quant à elle les corrélations pour le TBT. Sur ces figures, les flèches pleines montrent que le besoin de l'information d'awareness est plus important en fonction de la variable. Les flèches pointillées indiquent au contraire que l'information devient moins importante. Sur les flèches apparaissent des valeurs entre crochets, indiquant des conditions pour lesquelles l'importance évolue. Pour chaque corrélation, le niveau

de signification est indiqué par le nombre d'étoiles (*** pour 0,01, ** pour 0,05 et * pour 0,1).

Pour être plus explicite, je vais prendre un exemple. Dans le type de collaboration RT, nous pouvons voir qu'il existe une corrélation entre la *taille de l'équipe* et l'*Opinion*. Ce qui signifie que plus la taille de l'équipe est élevée, plus l'opinion est importante. Nous pouvons également voir que *Presence History* est plus important dans le cadre d'un projet étudiant.

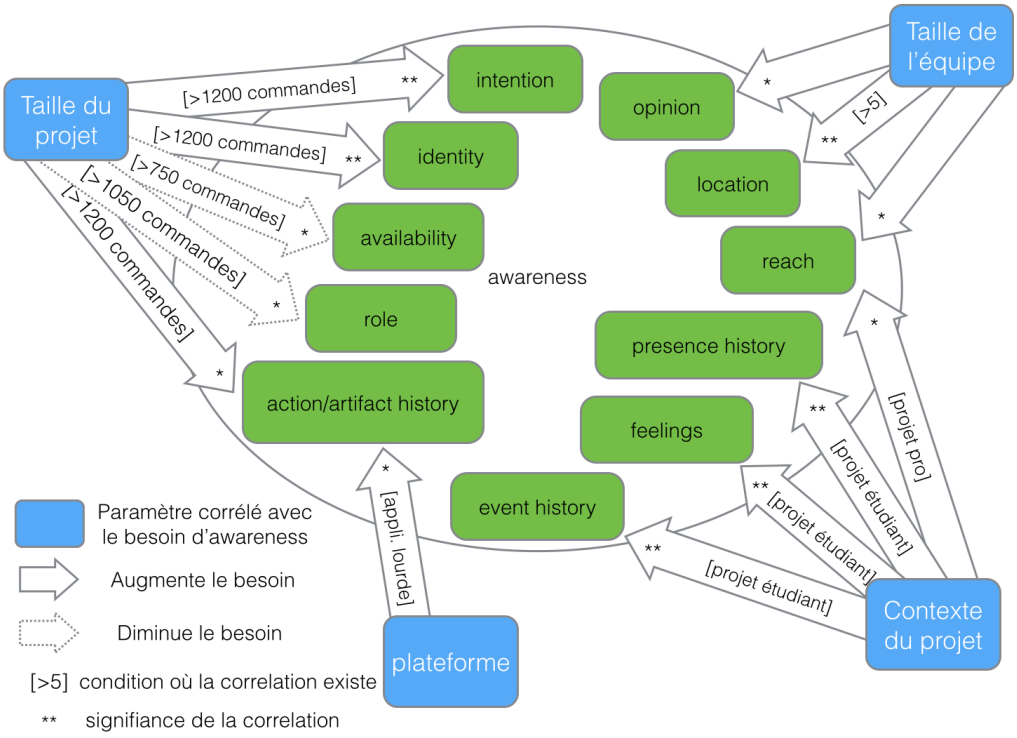


FIGURE 6.4 – Impact de paramètres sur les résultats pour le RT

Les tests de corrélation ont montré que l'utilisation de l'outil de modélisation n'influençait pas sur les résultats, nous pouvons donc généraliser ces résultats pour tous les outils de modélisation. Dans cette étude, nous avons pu voir que la distance entre les utilisateurs n'influençaient pas les résultats. Ces résultats ne sont donc pas applicables uniquement dans le cadre d'équipes réparties à travers la planète. Ils sont également valables pour des équipes partageant le même bureau.

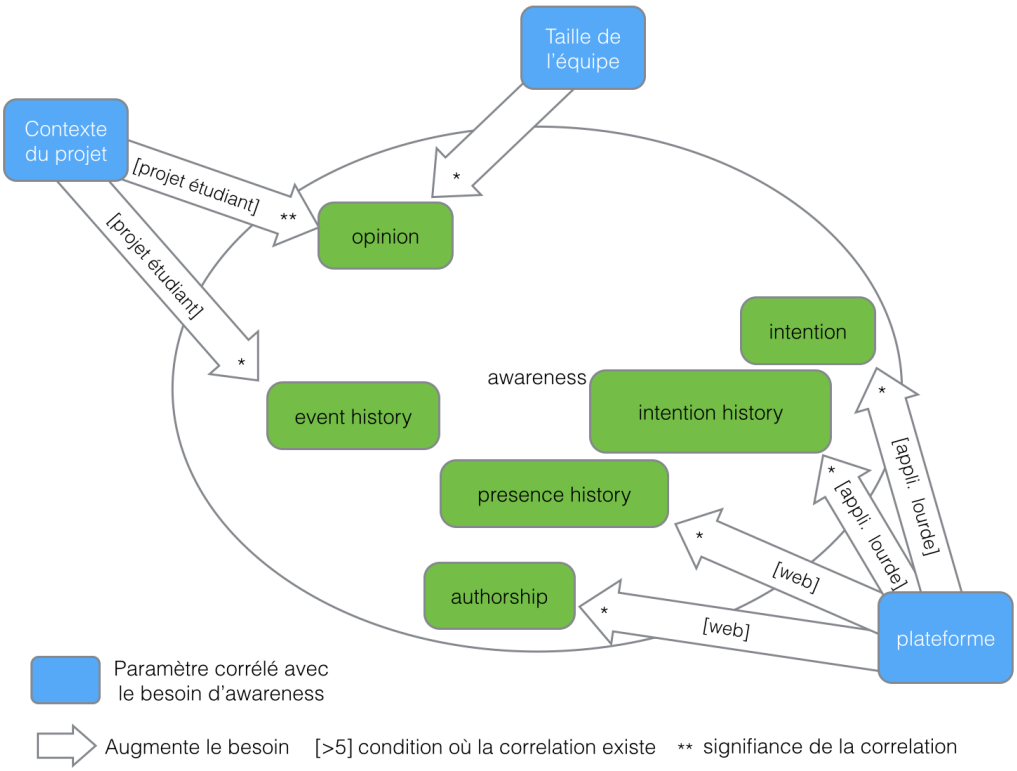


FIGURE 6.5 – Impact de paramètres sur les résultats pour le TBT

6.4 DISCUSSIONS

En comparant le support des informations d'awareness dans les outils existants et les résultats de l'étude présentée précédemment, nous constatons des écarts.

En effet, la plupart des fonctionnalités collaboratives permettent d'adresser les informations d'awareness les plus importantes. Néanmoins, certaines sont manquantes, en particulier pour le TBT où l'information Action/Artifact History n'est pas ou peu supportée. De plus, les outils ne prennent pas en compte les différents types de collaborations.

Amélioration possible dans les outils de modélisation
L'affichage des informations pourrait être différent selon le type de collaboration en cours. En fonction du mode utilisé, uniquement les informations d'awareness les plus importantes pourraient être affichées à l'utilisateur. L'affichage serait donc contextualisé.

L'opinion des collaborateurs est importante pour l'OC. Néanmoins, il n'est pas possible de consulter l'avis des autres dans les outils de collaboration excepté pour Creately.

Cette étude a permis de mettre en avant les informations d'awareness à afficher pour les différents types de collaboration. De plus, les informations relatives au passé, notamment celles concernant les changements réalisés sont extrêmement importantes. Bien que celles-ci soient très souvent oubliées dans les outils et dans les travaux, elles sont nécessaires aux utilisateurs.

7 Support basique de l'awareness

Nous venons de définir les informations d'awareness les plus importantes pour chaque type de collaboration. Celles-ci ne sont pas intégralement supportées d'après l'état de l'art. Les premières fonctionnalités implémentées, présentées dans ce chapitre ne présentent aucune difficulté conceptuelle ou technique et ne diffèrent pas des solutions présentées lors de l'état de l'art. Dans ce chapitre, je vais présenter la réalisation de ces fonctionnalités en détaillant comment l'information d'awareness est récupérée et traitée.

7.1 Gestion des droits

La fonction de gestion de droit est tout d'abord utile pour pouvoir partager le modèle entre plusieurs collaborateurs, raison pour laquelle elle est présentée ici. De ce point de vue, aucune information d'awareness n'est affichée. L'awareness entre en compte lorsque l'on ajoute des droits plus fins comme le droit de lecture, de commentaire, etc. Pour le moment, le gestionnaire de droits supporte quatre types de droits : les droits d'édition, de lecture uniquement, de lecture de versions majeures et d'administration comme l'illustre la figure 7.1. De ce fait, l'information d'awareness *Reach* est partiellement supportée de la même manière que nous avons pu le voir lors de l'état de l'art. Pour afficher pleinement cette information, il faudrait des droits plus fins et localisés. Par exemple, pour un utilisateur il serait possible de lui donner des droits de commentaires sur une vue uniquement plutôt que sur l'intégralité du projet.

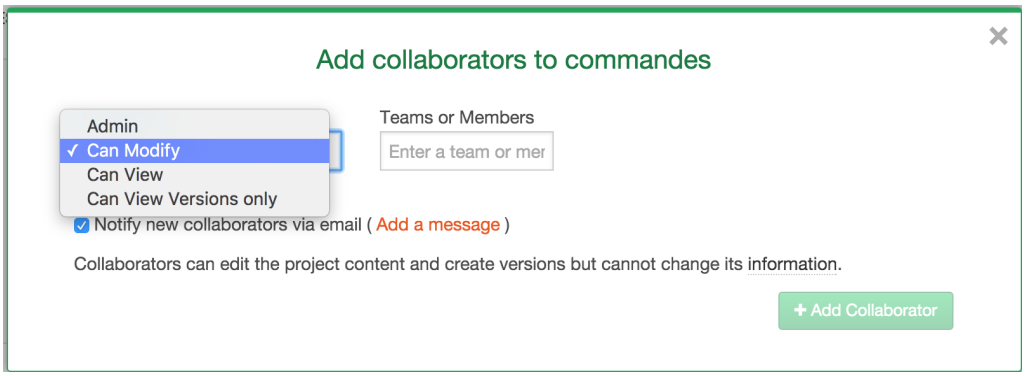


Figure 7.1 – Gestion des droits d'accès à un projet

7.2 Chat et collaborateurs connectés

Le chat sert à fournir un moyen de communication basique au sein de l'outil. La communication est importante et fait partie du modèle 3C. Il est pour l'instant basique mais des idées pour améliorer la communication seront présentées à la fin de cette thèse. La liste des collaborateurs

Figure 7.2 – Liste des utilisateurs connectés dans GenMyModel

connectés affiche les avatars des utilisateurs lorsqu'ils les ont définis ou un cercle contenant la première lettre de leur prénom comme pour la solution Google comme l'illustre la figure 7.2. Lors du passage du curseur de la souris sur les éléments, le nom complet de l'utilisateur est affiché. Cette fonctionnalité permet d'afficher les informations d'awareness *Presence* et *Identity*.

Récupération des informations d'awareness

Le framework CometD est utilisé pour les échanges de messages entre les clients et le serveur comme présenté dans la première partie de la thèse. Ce framework permet de savoir quand un utilisateur vient de rejoindre ou quitter une room. Lorsqu'un de ces événements est déclenché, un message est envoyé à tous les utilisateurs pour les notifier de l'arrivée ou du départ d'un utilisateur. Le message au format JSON contient l'identifiant du projet, ainsi que l'identifiant de l'utilisateur, ce message est ensuite envoyé à tous les utilisateurs par un bus spécifique. Il est de la forme suivante :

```
{user: userId, project: projectId, type: connection/
    disconnection}
```

Affichage des informations

Lorsqu'un utilisateur reçoit le message, il ajoute ou supprime l'utilisateur de la liste des collaborateurs connectés. Une API a été développée au sein de GenMyModel pour pouvoir récupérer des informations relatives aux projets et aux utilisateurs. Un appel à cette API est réalisé pour récupérer les informations de l'utilisateur dans le cas d'une connexion dans le but d'obtenir son avatar s'il est défini ainsi que son nom.

7.3 Focus

Le focus cible les informations *Action*, *Artifact* et *Location* en mettant en valeur les éléments de modèles sélectionnés par les utilisateurs comme illustré sur la figure 7.3. Les deux premières faisant partie des informations importantes à afficher, il était nécessaire d'ajouter cette fonctionnalité. En l'état actuel, l'affichage de l'information reste incomplet puisque la surbrillance de l'élément de modèle n'est visible aux utilisateurs que si l'action est réalisée dans leur zone d'affichage. En dehors, ils ne seront pas en mesure de les percevoir.

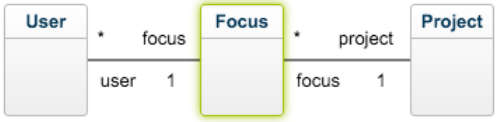


Figure 7.3 – Affichage du focus dans GenMyModel

Une solution envisageable pour palier ce manque pourrait être de mettre en surbrillance les noms des diagrammes sur lesquels d'autres utilisateurs sont en train d'effectuer des modifications par la couleur qui leur est attribuée.

Récupération des informations d'awareness

Les informations transmises lors du clic d'un utilisateur sur un élément de modèle concernent la personne ayant effectuée l'action, l'élément de modèle concerné et le projet. Le message transmis se présente alors sous la forme suivante :

```
{user: userId, project: projectId, element:
   graphicalModelElementId}
```

Il est nécessaire que l'information se rapportant à l'élément de modèle concerne sa représentation graphique. En effet, un même élément de modèle peut avoir plusieurs représentations graphiques. Dans le cas où le message contiendrait l'élément de modèle lui-même, il ne serait pas possible de déterminer sur quelle représentation l'utilisateur a cliqué. Techniquement, le message précédent est envoyé au format JSON également au serveur au travers d'un bus dédié de la même manière que pour les actions des utilisateurs présentées en première partie de cette thèse ainsi que pour les messages de chat. Le serveur renvoie ensuite le message JSON à tous les autres collaborateurs connectés.

Affichage des informations

Une fois le message JSON reçu par un client, il reste à mettre en valeur l'élément de modèle. Chaque collaborateur possède une couleur qui lui a été attribuée et visible dans le panel des utilisateurs connectés. A partir de l'identifiant de l'élément de modèle sélectionné, il suffit de retrouver cet élément de modèle pour ensuite lui appliquer un style CSS. J'ai choisi d'encadrer les éléments de modèles par la couleur attribuée à l'utilisateur. Dans le cas de liens entre éléments (associations, compositions...), la couleur du lien est changée.

7.4 Gestion de tâches

Nous souhaitons maintenant proposer un support pour la gestion des tâches dans le but d'adresser les informations d'awareness *Intention* et *Intention History*. Le focus ne permet pas de supporter pleinement l'information *Intention* puisque lorsqu'un utilisateur sélectionne un élément, il n'est pas possible de déterminer si l'action réalisée sera une suppression, un changement des propriétés de l'élément... L'ajout d'un système de tâches permet de palier ce problème. Par tâche, nous entendons la liste des étapes à réaliser pour aboutir au travail final. Une première version de la fonctionnalité a été mise en place au sein du chat. Il est ainsi possible d'afficher, de créer, de changer l'état des tâches et de les attribuer à des collaborateurs.

Récupération des informations d'awareness

Les informations sur les tâches à accomplir dépendent de la saisie des utilisateurs. Ceux-ci doivent la saisir en tapant le texte suivant comme le montre la figure 7.4 : #todo "*label*" [assignedUser]. Le texte précédent

crée une nouvelle tâche nommée *label* et peut éventuellement être directement attribuée à un collaborateur en saisissant son nom d'utilisateur. Un collaborateur fait part de son intention de commencer une tâche en saisissant le texte #start suivi de l'identifiant de la tâche. Cet identifiant est accessible à partir de la liste des tâches présentée ensuite.

Enfin un utilisateur fait part de la fin de la réalisation d'une tâche en saisissant le texte #end suivi de l'identifiant de la tâche.

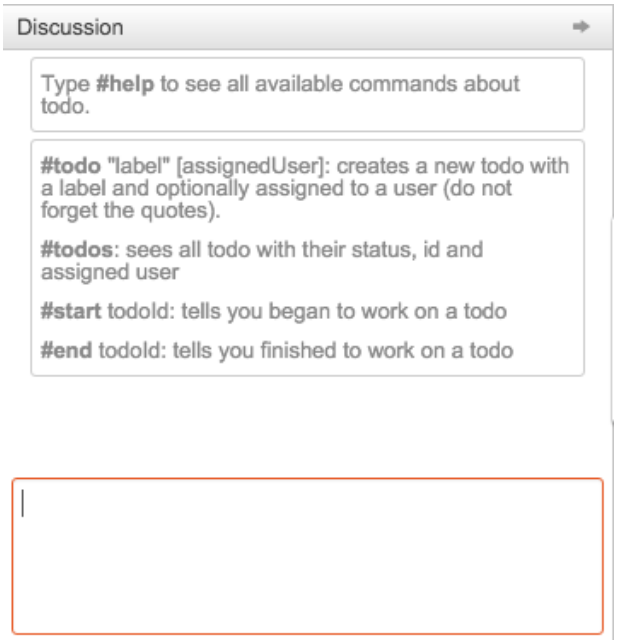


FIGURE 7.4 – Saisie de l'information *Intention* dans la fonctionnalité de gestion des tâches.

Affichage des informations

L'utilisateur est capable d'afficher la liste des tâches au moyen du texte suivant : #todos La figure 7.5 présente un scénario typique d'utilisation de la fonctionnalité. L'utilisateur *micheld* commence par récupérer la liste des tâches en cours (1) qui est vide pour le moment. Il crée ensuite une nouvelle tâche intitulée *Add user accounts* qu'il s'attribue. Un message de chat est alors posté pour signaler à tous les collaborateurs la création de cette nouvelle tâche (2). En listant à nouveau les tâches, il peut voir sa tâche avec l'état en cours placé à *TODO*, l'intitulé *Add user accounts*, l'id 93 et qu'elle lui est attribuée (3). Il commence ensuite à travailler sur cette tâche. Tous les collaborateurs sont notifiés du début de

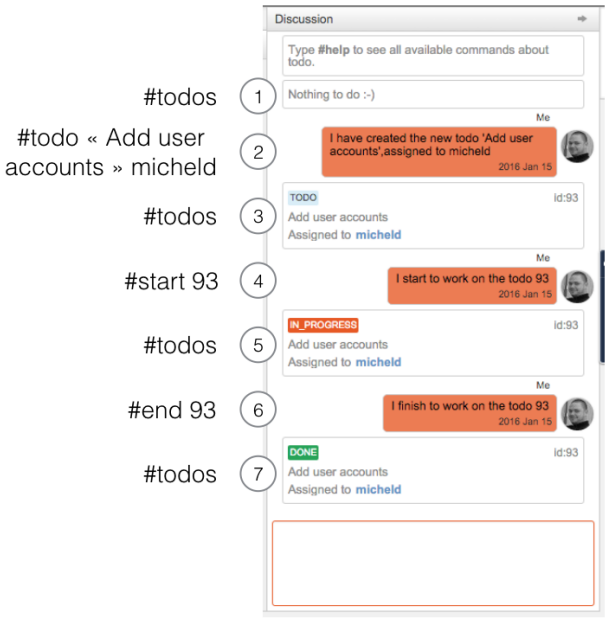


FIGURE 7.5 – Scénario d'utilisation de la fonctionnalité de gestion des tâches

l'activité par un message de chat (4). En listant les tâches, le statut de la tâche est passé de *TODO* à *IN_PROGRESS* (5). L'utilisateur a maintenant fini sa tâche et le saisit dans le chat. Un nouveau message est alors envoyé (6). En listant une dernière fois les tâches, la sienne est passée du status *IN_PROGRESS* à *DONE*. En listant les tâches, il est ainsi possible d'identifier rapidement quelles sont les tâches à réaliser, celles en cours et par qui elles sont réalisées ainsi que celles terminées.

Vue générale sur les premières fonctionnalités collaboratives

Pour coordonner ces fonctionnalités collaboratives, un système de données a été défini, notamment pour la sauvegarde des informations en base de données. La figure 7.6 illustre les différents éléments stockés en base de données en gris et en bleu les fonctionnalités collaboratives utilisant les informations. Nous pouvons y voir qu'un utilisateur (*User*) a l'accès à un projet (*Project*), grâce à des *AccessRule*. Celles-ci sont utilisées pour savoir si un utilisateur a le droit d'accéder à un projet et de quels droits il dispose. Ensuite, les utilisateurs s'échangent des *DiscussionMessage*, ils sont liés à un utilisateur sur un projet particulier. Ceux-ci sont affichés et créés dans le *ChatPanel*. Enfin, lorsqu'un utilisateur clique sur

un élément de modèle, un *FocusMessage* est défini et enregistré en base de données. Ce message lie un utilisateur à un projet et un élément de modèle, accessible par son identifiant unique appelé *ModelElementUUID*. Le *FocusMessage* est enregistré en base de données pour pouvoir donner ces informations à un utilisateur rejoignant la session de collaboration. Dès lors que l'utilisateur n'a plus le focus sur l'élément de modèle, le *FocusMessage* est supprimé.

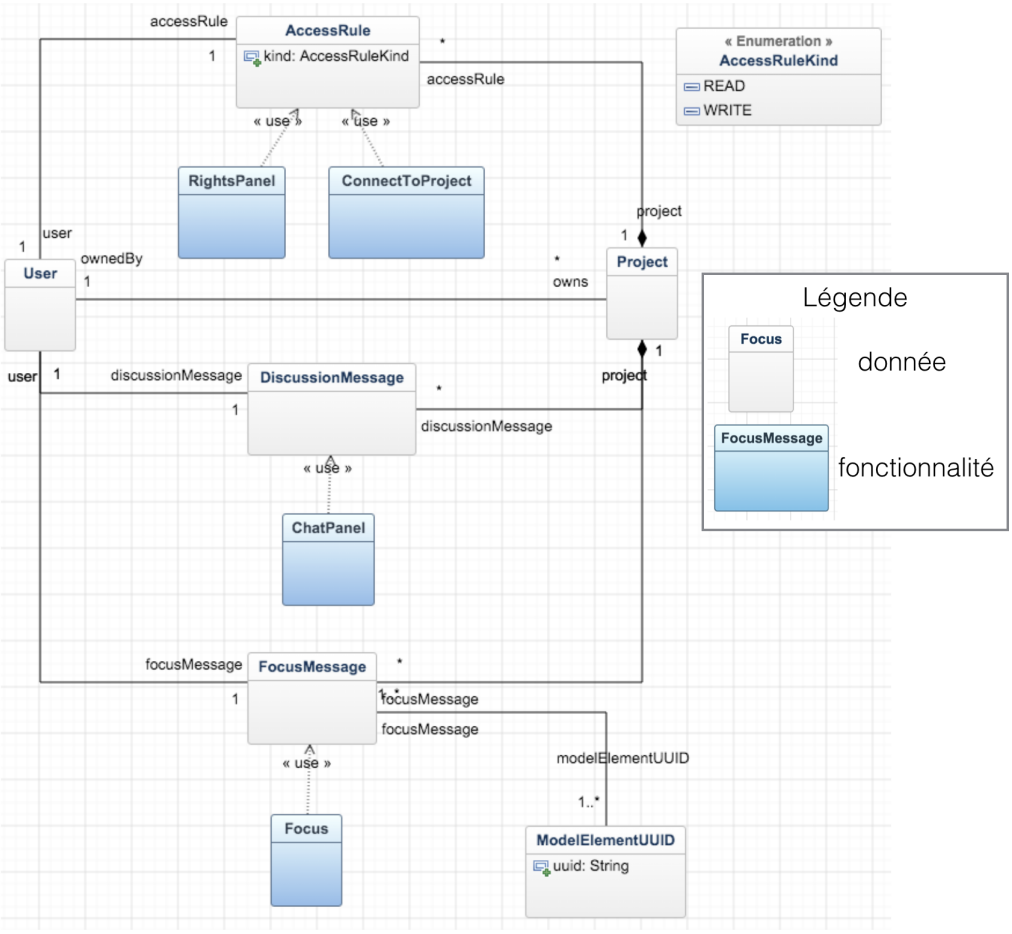


FIGURE 7.6 – Vue générale de l'implémentation des fonctionnalités collaboratives

Avec l'ajout de ces nouvelles fonctionnalités, trois nouveaux bus ont pris part dans le système d'échange de messages : un bus pour les messages de chat, un pour la connexion des utilisateurs et un dernier pour les focus. Le système présenté sur la figure 2.2 devient donc celui montré sur la figure 7.7.

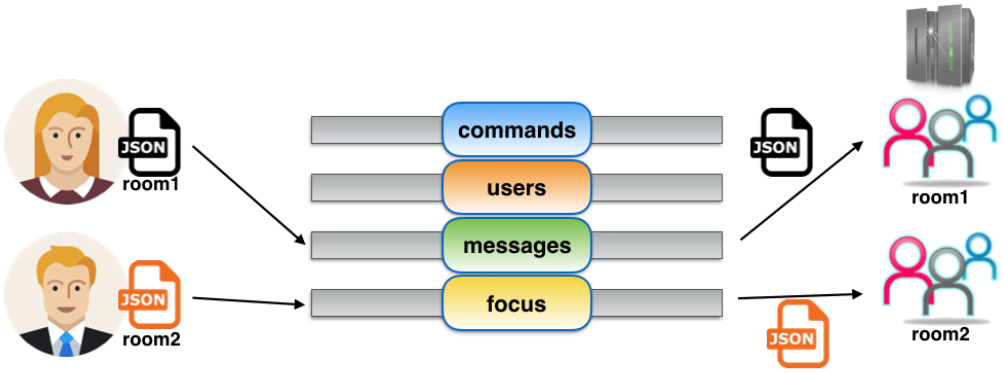


FIGURE 7.7 – Fonctionnement des rooms et des bus avec les fonctionnalités collaboratives

7.5 BILAN SUR CES PREMIÈRES FONCTIONNALITÉS

Ces premières fonctionnalités ont permis d'afficher les principales informations d'awareness aux utilisateurs.

La plupart de ces fonctionnalités n'ont rien d'innovant puisqu'elles proposent les mêmes principes que les outils et travaux présentés lors de l'état de l'art. Seule la fonctionnalité de gestion de tâches n'apparait pas dans les autres outils.

Pour certaines de ces fonctionnalités, il a été possible de quantifier leur utilisation. Par exemple, à la date du 18 Janvier 2016, soit plus d'un an après la mise en production des fonctionnalités chat et gestionnaire de tâches, il y a eu 26126 messages de chat et 93 tâches. Ces résultats ne sont pas satisfaisants. En effet, par rapport au nombre de projets collaboratifs créés (environ 20 000), le chat est très peu utilisé. Le gestionnaire de tâches dépendant du chat, son utilisation est donc encore plus faible, malgré le fait que cette fonctionnalité ait été promue au travers du blog [1]. J'avais déjà pu remarquer auparavant que le chat n'était que très peu utilisé. Dans un premier temps, je pensais que la fonctionnalité n'était pas assez visible pour les utilisateurs. Après avoir revu l'ergonomie, le souci est resté. Deux explications sont alors possibles pour justifier sa faible utilisation :

Le chat est inutile puisque les collaborateurs communiquent autrement.

La modélisation n'est qu'une étape dans le développement logiciel

1. http ://blog.genmymodel.com/discover-the-revision-history-in-your-genmymodel-projects.html

et l'outil de modélisation n'est qu'un des outils utilisés dans tout ce processus. Il parait donc naturel que les collaborateurs utilisent le même outil de communication durant tout le processus de développement et non celui de chaque outil qu'ils utilisent.

Le chat doit être amélioré pour aider les utilisateurs à communiquer plus efficacement. Par exemple, lorsque les utilisateurs discutent par rapport à certains éléments de modèle, il devrait être possible de mettre en surbrillance ces éléments sur le modèle grâce à l'utilisation de tags.

Le gestionnaire de tâches quant à lui pourrait devenir indépendant du chat pour plus de visibilité.

Après l'ajout de ces fonctionnalités, il reste néanmoins une information importante à afficher : Action History. Cette information est loin d'être répandue dans les outils et dans les travaux, surtout en modélisation. Pour afficher cette information, j'ai décidé d'implémenter une nouvelle fonctionnalité : un historique des changements.

8 Fonctionnalité d'historique

Dans le chapitre précédent, nous avons détaillé les premières fonctionnalités collaboratives implémentées permettant d'afficher une partie des informations les plus importantes. Les informations d'awareness restantes concernent celles relatives au passé. En effet, les fonctionnalités précédentes ne permettent pas d'afficher les informations *Action History* et *Artifact History*. La réponse couramment employée pour répondre à cette problématique est un historique des modifications puisque le but est d'afficher les modifications ayant eu lieu dans le passé et par conséquent connaitre aussi l'identité des auteurs de ces changements. Dans l'état de l'art sur les fonctionnalités collaboratives dans les outils de modélisation, nous avons pu voir que l'historique était présent dans peu d'outils. De plus, le fonctionnement de ces historiques ne permettent pas d'afficher complètement ces informations puisqu'ils n'affichent que des versions majeures. La suite de ma thèse s'est donc portée sur la réalisation de cet historique. Ce chapitre va donc présenter les modifications que j'ai dû apporter sur le système de commande, détaillé dans la première partie de cette thèse, pour ensuite montrer le fonctionnement de l'historique que j'ai implémenté.

8.1 Amélioration du système de commandes

La fonctionnalité d'historique nécessite les informations *Action History* et **Artifact History**. Ces informations consistent à ordonner dans le temps les actions ayant été réalisées. Ces dernières correspondent aux commandes définies dans le chapitre 3. Les commandes doivent donc être modifiées, en ajoutant une date pour ajouter la notion de temps. Il faut également ajouter l'auteur de la commande pour pouvoir afficher l'information d'awareness *Authorship*, importante d'après l'étude réalisée. Pour résumer, sur chaque commande, les informations suivantes ont été ajoutées :

user : l'utilisateur étant à l'origine du déclenchement de la commande.

date : date à laquelle le *user* a créé la commande.

8.2 Présentation de l'historique

Pour présenter l'historique de manière originale, j'ai choisi de le rejouer comme une vidéo. L'enchainement des commandes se fait au travers du temps et l'utilisateur doit identifier les changements réalisés

entre chaque action. Pour ce faire, la vidéo me semblait être la meilleure solution. Néanmoins cette solution n'existe pas dans les outils de modélisation actuellement mais elle a été rencontrée dans un IDE en ligne : Cloud9. Même si la fonctionnalité existe déjà dans l'IDE, il n'est pas possible de l'appliquer de la même manière dans les outils de modélisation. Dans notre outil, chaque action de l'utilisateur correspond à une commande alors que dans les IDE, une commande n'est pas associée à chaque saisie de texte d'un utilisateur. La granularité des informations est différente entre les deux systèmes.

L'historique est présenté dans sa première version sur la figure 8.1 et fut mise à disposition des utilisateurs de GenMyModel le 9 février 2015. La figure présente l'historique une fois le chargement terminé. A partir

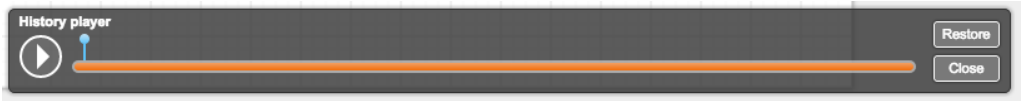


FIGURE 8.1 – Première version de l'historique

de cet état, l'utilisateur peut cliquer sur la barre de progression pour visualiser un état précis du modèle. Il peut aussi juste passer le curseur de la souris sur la barre pour avoir plus d'informations sur une version comme le montre la figure 8.2. Lors du passage du curseur, l'utilisateur peut voir la date et l'auteur de la commande ainsi que le label détaillant celle-ci.

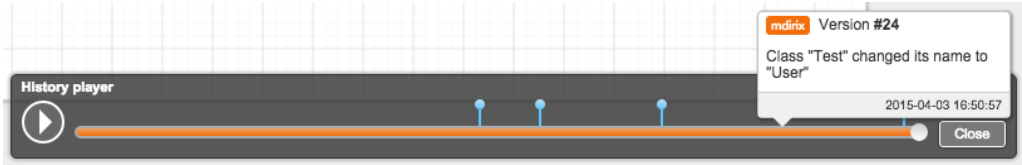


FIGURE 8.2 – Informations complémentaires sur une version de l'historique

8.3 IMPLÉMENTATION

L'objectif de notre historique est de pouvoir afficher à l'utilisateur toutes les actions ayant eu lieu sur le modèle. De ce fait, il faut se placer dans les mêmes conditions d'exécution que lors de la phase de création du modèle. Cette exécution se fait au travers de commandes EMF

comme présenté dans la première partie de cette thèse. Nous repartons donc d'un modèle vide puis nous récupérons toutes les commandes au travers de l'API, présentée auparavant. Une fois toutes récupérées côté client, celles-ci sont converties dans des commandes EMF de la manière que dans la première partie de la thèse. La différence avec l'exécution en temps-réel des commandes est que lors du chargement de l'historique, nous n'exécutons pas directement les commandes sur le modèle mais au travers d'une *stack* EMF. Cette stack permet de garder en mémoire l'exécution des commandes réalisées sur le modèle. Une première phase exécute donc rapidement toutes les commandes sur cette *Stack* pour ensuite donner la main à l'utilisateur pour les parcourir. L'utilisateur voit ainsi la dernière version du modèle et la *Stack* contient toutes les commandes. Pour pouvoir naviguer dans les changements, j'utilise une des possibilités de la *Stack* à savoir le fait d'annuler ou de rejouer la dernière commande (*undo/redo*). Par exemple, si l'utilisateur se place à la version 15 du modèle et qu'il veut maintenant voir la version 10, 5 *undo* seront réalisés sur la *Stack*.

8.4 Validation de l'historique

La manière d'afficher l'historique des changements dans les outils de modélisation étant nouvelle, une première validation exploratoire fut réalisée. Cette première étude avait pour but de déterminer si l'historique était bien utilisé pour afficher les bonnes informations d'awareness mais aussi de déterminer les cas d'utilisation de la fonctionnalité. Une seconde étude fut mise en place pour valider le design et l'usage de l'historique mais aussi pour approfondir les résultats de la première étude. La suite de ce chapitre détaillera les différentes études réalisées ayant pour but de déterminer les cas d'usage de l'historique ainsi que le ressenti des utilisateurs.

8.5 Etude exploratoire

Description

Cette première étude fut réalisée auprès d'étudiants durant l'un de mes enseignements. Ces étudiants étaient des Master Miage 1ère année et je les ai suivis durant un semestre dans un module de gestion de projet. Dans ce module, ils devaient réaliser leur projet en mode start-up, c'est-à-dire trouver une idée innovante et la tester auprès de réels utilisateurs par petites itérations de manière à impliquer les utilisateurs finaux

dans le processus de développement. Lors de la mise en place de leur projet, je leur ai demandé de modéliser leur application et ensuite tenir à jour leur modèle en fonction des modifications réalisées sur le code. Ces applications étaient de types complètement différents :

— Application mobile de gestion de séances de musculation.

— Application web d'aide à la création de tutoriels.

— Application mobile d'aide et location de places de parking.

— Application mobile de revue de jeux vidéos.

— Gilet sensoriel pour les jeux vidéos.

Chaque groupe réalisant un projet était constitué d'au moins 3 étudiants. L'historique a été mis en production vers la fin de leur projet. J'ai demandé aux étudiants de tester l'historique et j'ai ensuite réalisé des interviews avec eux pour recueillir leur ressenti face à cette fonctionnalité mais aussi déterminer les cas d'utilisation et les améliorations possibles. Cette première étude avait surtout pour but de déterminer si l'utilisation de l'historique était celle attendue. Pour cela l'interview était semi-guidée. Parmi les questions posées, je demandais :

— pourquoi ils utilisaient l'historique ou pourquoi ils pourraient l'utiliser.

— s'ils ont vu qu'ils pouvaient déplacer la barre de navigation, les raccourcis clavier ou le survol sur la barre pour avoir plus d'informations.

— quelles pouvaient être les améliorations à apporter.

Résultats

Utilisation de l'historique

L'interview s'est portée sur 13 de ces étudiants. La durée totale de nos interviews était de 1h40. Chaque interview a été réalisée en face à face et a été enregistrée, réécoutée plusieurs fois de manière à synthétiser les réponses importantes. 4 réponses ont été données concernant la raison de l'utilisation de l'historique :

— Voir les changements réalisés par les autres.

— Quand des changements ont été réalisés et par qui.

— Restaurer une ancienne version

— Comprendre pourquoi les changements ont été réalisés.

La figure 8.3 montre le pourcentage de réponses. 100% des personnes interviewées ont utilisé l'historique pour voir les changements réalisés par les autres personnes de leur équipe. Les changements intéressants pour eux sont ceux réalisés après leur déconnexion. Cette réponse est intéressante pour deux raisons. Tout d'abord, l'usage que l'historique est bien celui attendu. En souhaitant connaitre les modifications réalisées par les autres personnes de l'équipe, ils souhaitent connaitre l'information d'awareness *Action/Artifact History*.

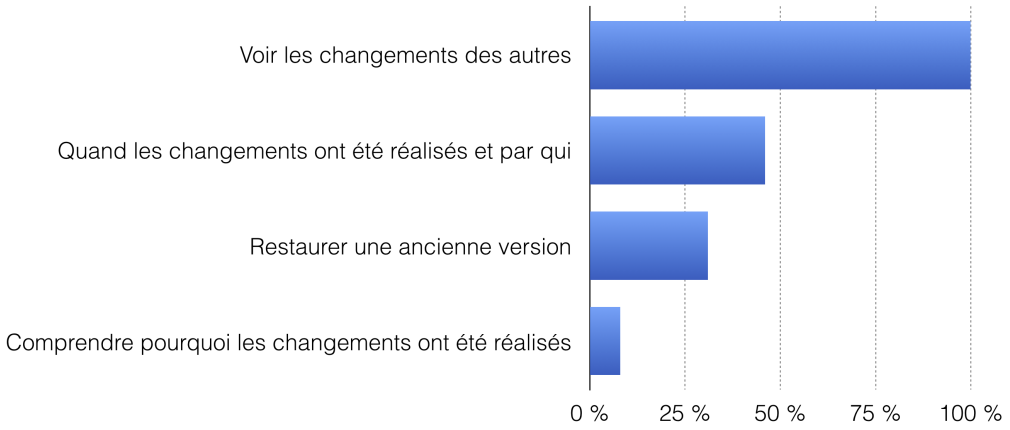


FIGURE 8.3 – Réponses sur l'utilisation de l'historique lors de l'interview exploratoire

Ensuite, la réponse pourrait confirmer une remarque faite lors de l'analyse des sessions des utilisateurs permettant de déterminer les différents types de collaboration. Lors de cette analyse, j'avais remarqué que le temps entre la connexion de l'utilisateur et sa première action était élevé. Ce temps pourrait être dû à la remise en contexte de l'utilisateur. Sans la présence de l'historique, il doit identifier les changements ayant eu lieu depuis sa dernière connexion. Avec l'historique, il peut identifier les changements réalisés depuis.

La seconde réponse la plus citée concerne la date des changements et l'auteur de ces changements. Encore une fois, ces informations sont relatives aux informations d'awareness *Action/Artifact History* et *Authorship*.

Les questions posées sur l'usage de l'historique semblent confirmer que l'historique répond bien au besoin de départ. Néanmoins, le nombre de personnes étant faible, il n'est pas possible de tirer de conclusions précises à ce sujet.

Visibilité des fonctionnalités

La seconde partie de l'interview portait sur le design et l'ergonomie de trois fonctionnalités de l'historique en leur demandant s'ils avaient remarqué qu'ils pouvaient :

— Déplacer de la barre de progression afin de visualiser une modification particulière.
— Utiliser des raccourcis clavier pour naviguer dans les changements.
— Survoler la barre de progression pour visualiser plus d'informations comme l'auteur du changement ou la date.

Cette partie s'est révélée surprenante... Comme le montre la figure 8.4, les raccourcis clavier n'ont été vus par aucune des personnes interviewées. GenMyModel comprend une entrée dans les menus permettant d'accéder aux raccourcis clavier de toute l'application mais les personnes interviewés ne l'ont jamais ouvert. La plupart des interviewés m'ont fait part de la pénibilité de la navigation dans les modifications lors d'un parcours à fin grain. Une grosse amélioration de l'historique devra donc se porter sur cette navigation à fin grain. Uniquement fin grain puisque la navigation dans la barre de progression (permettant de naviguer à gros grain) a été remarquée dans 85% des cas. Le survol de la barre quant à lui est visible pour 31% des interviewés, ce qui est très peu. Le survol n'étant pas intuitif, les informations concernant l'auteur de la modification ainsi que la date devront être affichées d'une autre manière.

Améliorations

La dernière partie de l'interview portait sur les améliorations possibles de l'historique. Il était alors demandé aux étudiants de donner leurs idées d'amélioration de l'historique. Trois grandes idées sont ressorties de ces discussions. La première consiste à afficher l'endroit où se situe leur dernière action sur la barre de progression. De cette manière, ils peuvent rapidement voir si beaucoup de modifications se sont produites depuis leur dernière déconnexion. Pour cela, un marqueur pourrait être apposé sur cette barre.

La seconde amélioration réside dans l'ajout de boutons *suivant* et *précédent* pour une navigation plus précise. Les raccourcis étaient présents mais non visibles, des boutons doivent être présents pour effectuer ces actions.

Enfin, les étudiants souhaitaient voir plus rapidement le nom des auteurs des changements sans avoir à survoler la barre de progression pour obtenir cette information.

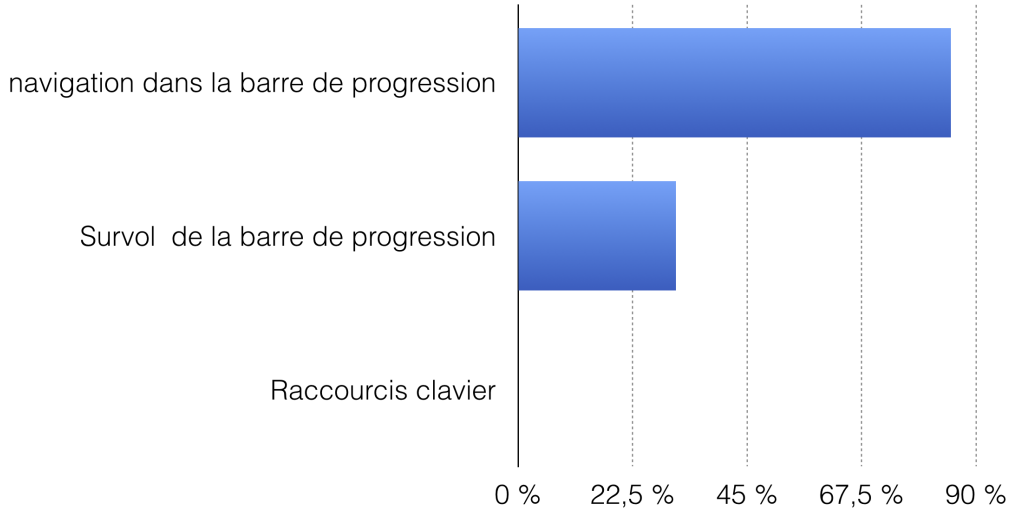


FIGURE 8.4 – Réponses sur la perception des fonctionnalités de l'historique lors de l'interview exploratoire

Morale de la première étude

Cette étude exploratoire a permis de mettre en avant différents points, positifs comme négatifs. Le gros point positif de cette étude fut les résultats concernant l'utilisation de l'historique, conforme aux attentes et répondant donc aux besoins des utilisateurs. Néanmoins les autres points de l'étude ont mis en avant les faiblesses de l'historique, notamment la visibilité des sous-fonctionnalités de l'historique et le confort d'utilisation.

8.6 MODIFICATIONS APPORTÉES

Les résultats de la première étude ont mis en avant un gros manque de visibilité de certaines fonctionnalités de l'historique, remettant en cause le design et l'ergonomie de celui-ci. Pour aider l'utilisateur dans la navigation fin grain, des boutons "suivant" et "précédent" ont été ajoutés avec la présentation au survol du raccourci clavier à utiliser pour naviguer commande par commande. Ces boutons sont visibles sur la figure 8.5. L'auteur de la commande étant une information importante, celle-ci est désormais visible tout le temps dans l'historique pour ne plus avoir à survoler la barre pour voir l'auteur de la commande en cours.

Ayant tiré des leçons de cette première expérimentation, une nouvelle validation a été réalisée en suivant cette fois-ci un protocole précis, basé sur l'UX. Plutôt que de nous limiter aux aspects IHM, j'ai effectivement

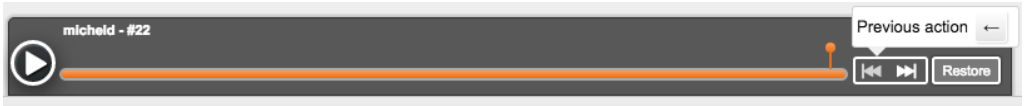


FIGURE 8.5 – Seconde version de l'historique

voulu effectuer une étude plus large en prenant en compte des aspects comme le ressenti tels que définis dans l'expérience utilisateur (UX) : " momentary, primarily evaluative feeling (good-bad) while interacting with a product or service"[Hassenzahl, 2008].

8.7 PRÉREQUIS SUR L'EXPÉRIENCE UTILISATEUR (UX)

D'après la norme ISO 9241-210, l'expérience utilisateur (UX) correspond aux réponses et aux perceptions d'une personne qui résultent de l'usage ou de l'anticipation de l'usage d'un produit, d'un service ou d'un système. (dans la suite du manuscrit, le mot produit englobera produit, service et système). Une partie de la définition pose problème à la communauté Human-Computer Interaction (HCI) : l'anticipation de l'usage d'un produit. En effet, cette notion implique les émotions des utilisateurs, difficilement anticipables. De ce fait, il n'existe pas encore de définition sur l'expérience utilisateur reconnue par l'ensemble de la communauté[Kujala et al., 2011].

Néanmoins, le modèle de l'UX proposé par Hassenzahl [Hassenzahl, 2005] est largement utilisé dans la recherche en UX. Celui-ci présente l'UX selon deux perspectives : la perspective du concepteur et la perspective de l'utilisateur. Le concepteur choisit et combine certaines caractéristiques pour le produit (contenu, présentation, fonctionnalités...) de manière à donner un caractère particulier au produit comme la nouveauté ou encore l'utilité. Néanmoins, ce caractère est subjectif car dépendant de l'intention du concepteur. Il peut être perçu différemment par les utilisateurs.

Lors de la prise en main d'un produit par un utilisateur, celui-ci en perçoit les caractéristiques et s'en construit une vision personnelle appelée *caractère apparent du produit*. Ce caractère apparent résulte des caractéristiques, des attentes et des habitudes des utilisateurs et produit des conséquences de trois types :

— un jugement sur l'attrait du produit.

— des émotions comme le plaisir, la satisfaction ou la déception.

— des comportements comme une utilisation accrue du produit.

D'après le modèle de Hassenzahl, les utilisateurs perçoivent les produits interactifs selon deux dimensions : les qualités pragmatiques et les qualités hédoniques[Hassenzahl et al., 2003].

Les qualités pragmatiques donnent la capacité à soutenir l'accomplissement des tâches. Un produit perçu comme ayant de bonnes qualités pragmatiques sera structuré, clair, contrôlable, efficace, pratique, etc.

Les qualités hédoniques donnent des explications sur le pourquoi de l'utilisation d'un produit. Un produit perçu comme ayant de bonnes qualités hédoniques sera original, créatif, captivant, présentable, professionnel, de bon goût, qui me rapproche des autres.

Dans le but de mesurer ces qualités, Hassenzahl a défini un questionnaire nommé Attrakdiff. Ce questionnaire est constitué de 28 paires de mots opposés et divisé en 4 catégories pour mesurer les qualités décrites précédemment :

Echelle de Qualité Pragmatique (QP) : Décrit l'utilisabilité du produit et indique le niveau de facilité perçue des utilisateurs à atteindre leur but.

Echelle de Qualité Hédonique - Stimulation (QHS) : Indique dans quelle mesure le produit soutient le besoin de stimulation en proposant des contenus, fonctionnalités, styles d'interaction nouveaux, intéressants et stimulants.

Echelle de Qualité Hédonique - Identité (QHI) : Indique dans quelle mesure le produit soutient une fonction sociale et communique une certaine identité de l'utilisateur.

Echelle d'Attractivité Globale (ATT) : Décrit la valeur globale perçue du produit basée sur la perception des qualités pragmatiques et hédoniques.

A l'origine, ce questionnaire fut établi en Allemand par Hassenzahl. Il fut ensuite traduit en Anglais, puis en Français par Lallemand [Lallemand et al., 2015]. Mon questionnaire s'adressant à des étudiants francophones, c'est cette dernière version que j'ai utilisée. Dans cette version, les paires de mots sont les suivantes :

— Humain/Technique

— M'isole/Me sociabilise

— Plaisant/Déplaisant

— Original/Conventionnel

— Simple/Compliqué
— Professionnel/Amateur
— Laid/Beau
— Pratique/Pas pratique
— Agréable/Désagréable
— Fastidieux/Efficace
— De bon goût/De mauvais goût
— Prévisible/Imprévisible
— Bas de gamme/Haut de gamme
— M'exclut/M'intègre
— Me rapproche des autres/Me sépare des autres
— Non présentable/Présentable
— Rebutant/Attirant
— Sans imagination/Créatif
— Bon/Mauvais
— Confus/Clair
— Repoussant/Attrayant
— Audacieux/Prudent
— Novateur/Conservateur
— Ennuyeux/Captivant
— Peu exigeant/Challenging
— Motivant/Décourageant
— Nouveau/Commun
— Incontrôlable/Maîtrisable

Chaque paire de mot est évaluée par l'utilisateur sur une échelle de Likert en 7 points et caractérise la qualité pragmatique ou hédonique. De cette manière, un score entre -3 et 3 est attribué pour chaque réponse, ce qui permet de classer le produit dans une des cinq catégories possibles comme le montre la figure 8.6.

Lorsqu'un produit est caractérisé par des qualités hédoniques et pragmatiques faibles, il est défini comme **Superflu**. Il est donc non désiré puisqu'il ne répond ni aux besoins pragmatiques, ni aux besoins hédoniques de l'utilisateur.

A l'inverse, le concepteur et l'utilisateur seront satisfaits lorsque le produit sera défini comme **Désiré**, puisque le produit répond bien aux qualités attendues par les deux partis.

Néanmoins, bien généralement, un produit se situe dans les caractéristiques intermédiaires où les deux qualités ne sont pas équilibrés. Pour ceux là, Hassenzahl a défini les produits **Trop orienté vers les tâches** et **Trop orienté vers soi**. Un produit trop orienté vers les tâches est un produit principalement pragmatique, ce qui signifie qu'il sera lié aux objectifs et tâches de l'utilisateur. L'attrait d'un tel produit dépendra donc de l'importance de l'objectif à atteindre par l'utilisateur. Un produit trop orienté vers soi quant à lui est lié à l'utilisateur lui-même puisqu'il est principalement hédonique. Un tel produit est plus facile à faire évoluer qu'un produit orienté vers les tâches puisqu'il est plus facile de faire évoluer l'appréciation du produit que l'importance des objectifs et des tâches de l'utilisateur.

8.8 Description de la validation de la seconde étude

Comme les principaux problèmes de la fonction historique portaient sur des problèmes de design et d'ergonomie, nous avons donc, après avoir effectué des évolutions, évalué l'historique selon une démarche de type UX, comme présenté juste au dessus. Notre évaluation s'est faite au travers d'un questionnaire distribué à des étudiants. Celle-ci se compose de trois parties. L'intégralité du questionnaire est présentée dans l'annexe A. La première partie du questionnaire est une application du questionnaire où les utilisateurs devaient noter chaque paire de mots sur une échelle de Likert de 7 points comme présenté par Hassenzahl. La seconde partie s'intéresse aux cas d'usage de l'historique afin de déterminer à quels moments l'historique est le plus utilisé et pour quelles raisons. Aux cas d'usage, des questions ont été ajoutées dans une troisième partie sur les actions importantes que l'historique doit mettre en valeur ainsi que la manière d'afficher ces actions dans le but d'améliorer la fonctionnalité. Cette seconde étude a été réalisée avec des étudiants en Licence et Master Informatique et MIAGE de l'Université de Lille. Ces étudiants ont été choisis puisque l'Université est un des clients importantes de GenMyModel et sont donc représentatifs du panel des utilisateurs. L'étude a été effectuée en deux temps. Le questionnaire a d'abord été soumis aux étudiants après la première utilisation de l'historique de manière à avoir leurs impressions sans être influencé par le reste de l'outil. Après trois mois d'utilisation, le questionnaire a été soumis à nouveau mais en gardant uniquement la partie usages.

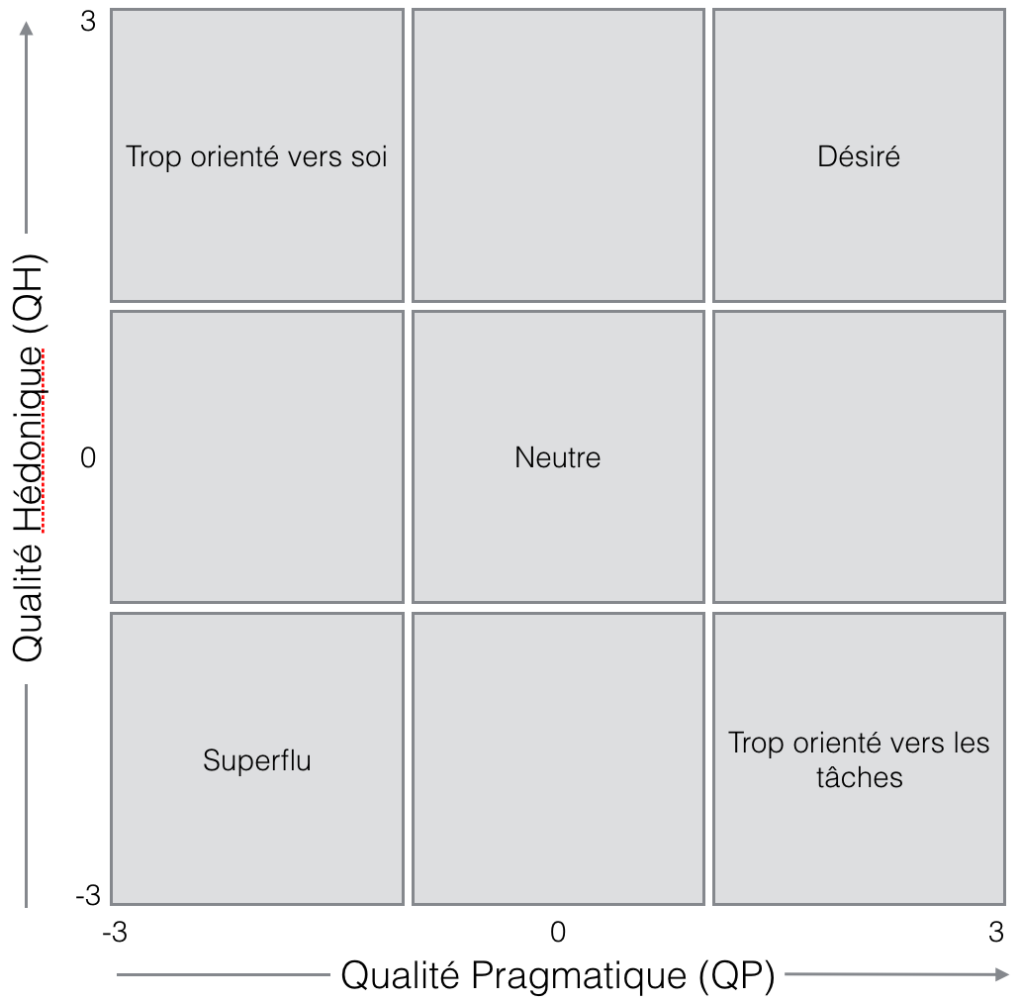


FIGURE 8.6 – Caractère d'un produit en fonction de ses qualités hédoniques et pragmatiques

8.9 VALIDATION DE L'HISTORIQUE DÈS LA PREMIÈRE UTILISATION

37 étudiants ont participé à la première évaluation. Le panel se compose de 5 femmes et 32 hommes âgés de 20 à 45 ans dont l'âge moyen est de 23 ans. Tous les étudiants sont en Licence et Master en Informatique et MIAGE et sont en moyenne à 4 par projet. Les étudiants avaient le choix entre deux projets : la diffusion d'annonces sur des écrans et la gestion d'une association pour du paintball adapté aux personnes à mobilité réduite.

AttrakDiff

L'évaluation des différentes paires de mots est présentée dans la figure 8.7. La note apparaissant pour chaque paire de mots est la moyenne attribuée à celle-ci. Les résultats sont très positifs. Néanmoins, quelques éléments tendent à être améliorés, notamment *Imprévisible/Prévisible* et *Peu exigeant/Challenging*. Ces résultats un peu en dessous des autres sont dus à quelques bugs rencontrés par les étudiants lors de l'étude. Ces bugs venaient de l'ajout d'informations sur le système de commande présenté en première partie (l'ajout de labels permettant de décrire l'action), et perturbant l'exécution de l'historique.

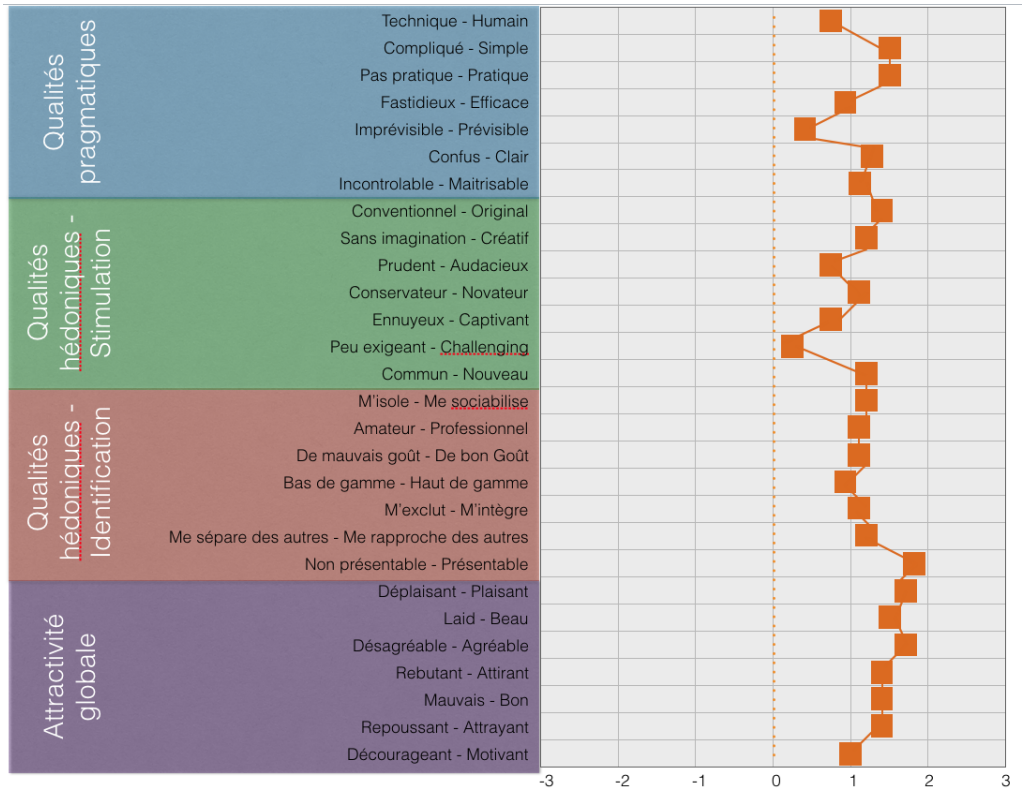


FIGURE 8.7 – Evaluation des paires de mots de l'AttrakDiff lors de la première utilisation de l'historique

Avant de classer la fonctionnalité, il est nécessaire de calculer la moyenne attribuée pour chaque dimension de l'AttrakDiff. Cette moyenne est calculée à partir de la moyenne attribuée à chaque paire de mots. Les qualités pragmatiques ont une moyenne de 1, les qualités hédoniques centrées sur la stimulation ont une moyenne de 0,9, celles sur l'identité

1,2. Enfin l'attractivité est évaluée à 1,4. Ces résultats sont présentés sur le graphique de la figure 8.8.

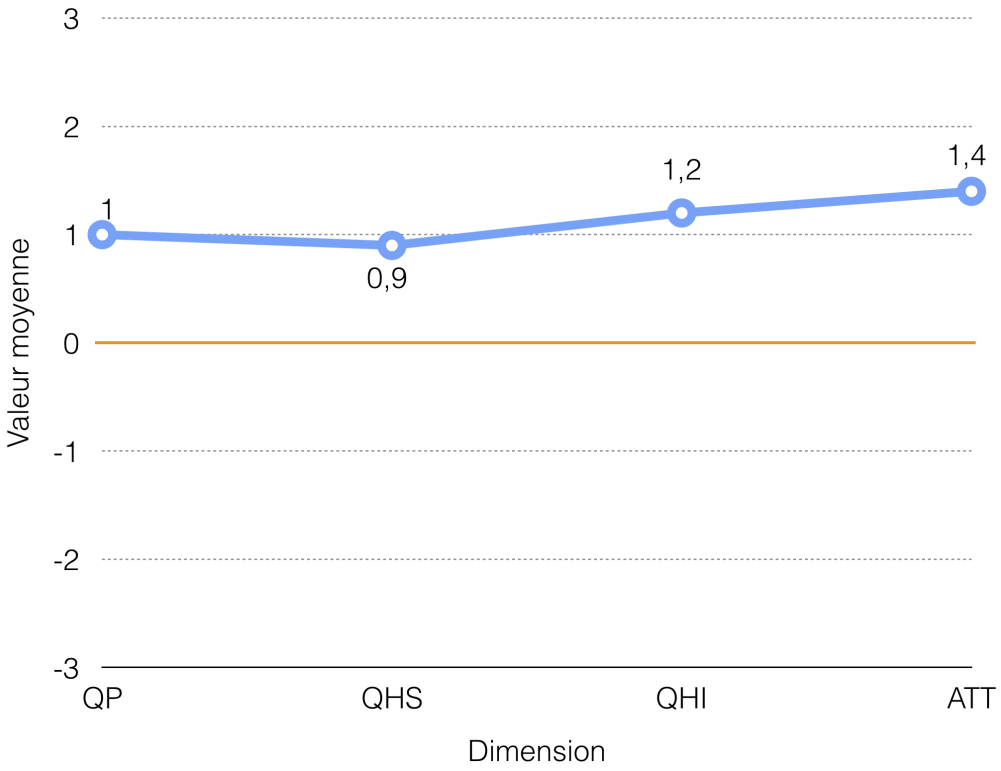


FIGURE 8.8 – Evaluation des dimensions de l'AttrakDiff lors de la première utilisation de l'historique

Une fois ces moyennes calculées, il est ainsi possible de catégoriser l'historique comme le montre la figure 8.9. L'historique est ainsi classé comme désiré. L'intervalle de confiance est quant à lui calculé à partir des écarts types des qualités hédoniques et pragmatiques. Un petit intervalle de confiance montre une homogénéité des résultats et donc une fiabilité et une précision des résultats, ce qui est le cas ici.

Usages

Cette partie de l'étude a pour but de vérifier que l'utilisation de l'historique est conforme aux usages ciblés à savoir fournir les informations d'awareness *Action History* et *Authorship* aux utilisateurs. Sur les 37 étudiants interrogés, 35 ont répondu à cette partie du questionnaire.

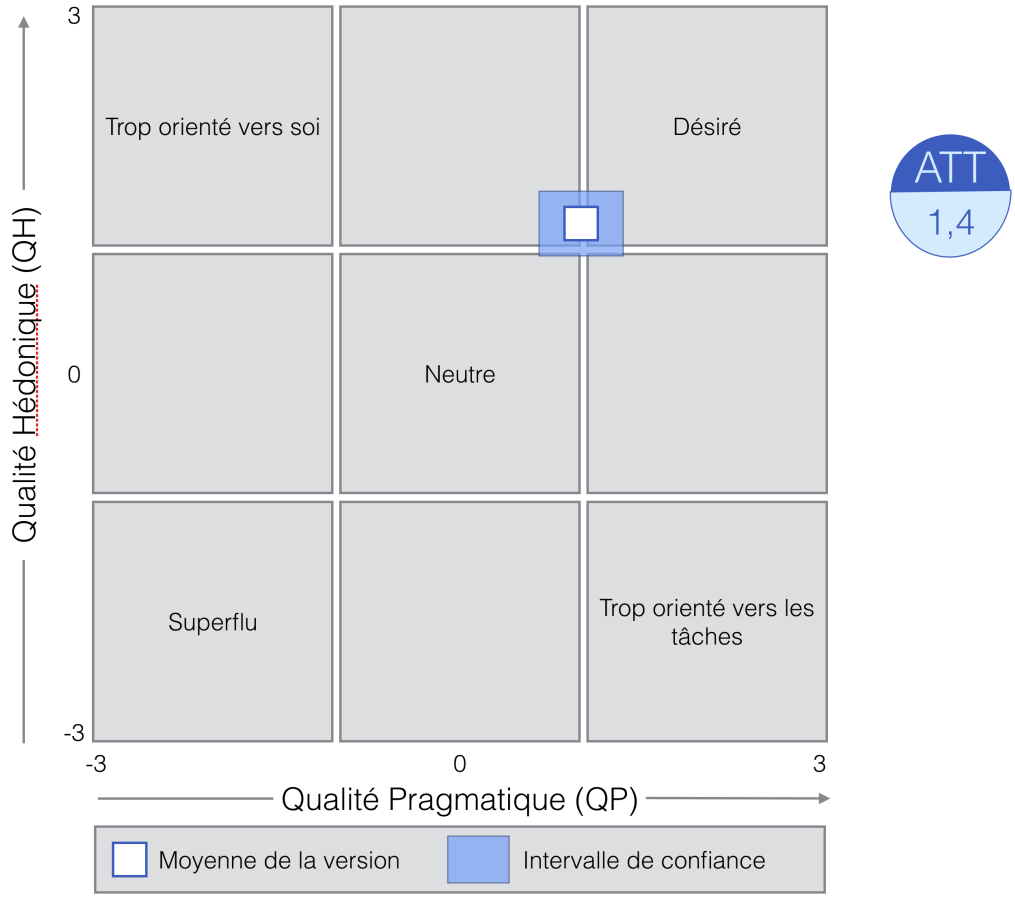


FIGURE 8.9 – Classement de l'historique selon les qualités hédoniques et pragmatiques de l'AttrakDiff lors de la première utilisation de l'historique

Pour déterminer les usages de l'historique, la question suivante a été posée, basée sur les résultats de l'étude exploratoire : Quelle(s) information(s) recherchiez-vous lorsque vous avez utilisé l'historique ?

- ☐ Les changements réalisés par les autres en particulier :
  - ☐ les suppressions
  - ☐ les ajouts
  - ☐ les modifications
  - ☐ tous les changements ayant eu lieu après ma déconnexion
  - ☐ autre
- ☐ Qui a réalisé des changements et quand en particulier :

- ☐ voir s'il y a eu des modifications après ma déconnexion
- ☐ pouvoir demander des informations à la personne concernée
- ☐ autre

☐ Restaurer une ancienne version en particulier :

- ☐ annuler des modifications faites par d'autres
- ☐ j'ai testé une solution et je reviens en arrière
- ☐ autre

☐ Pourquoi des changements ont été réalisés

☐ Je ne cherchais rien, j'ai été obligé de l'utiliser

☐ Autre raison

Les résultats de cette partie de l'étude sont présentés dans le tableau 8.1.

TABLE 8.1 – Part de réponse des usages de l'historique

Usage de l'historique	Part de réponse (%)
Changements réalisés par les autres	80
Suppressions	9
Ajouts	23
Modifications	34
Toutes depuis déconnexion	49
Autre	6
Qui a réalisé les changements et quand	57
Voir les modifications depuis déconnexion	40
Pouvoir demander des informations	23
Autre	3
Restaurer une ancienne version	43
Annuler les modifications des autres	26
Tester une solution	23
Autre	0
Pourquoi des changements ont été réalisés	17
Obligation de l'utiliser	11
Autre	14

Les résultats montrent que l'historique est largement utilisé pour voir les changements réalisés par les autres (80%), notamment pour suivre toutes les modifications ayant eu lieu depuis la déconnexion de l'utilisateur dans la moitié des réponses. Cette réponse confirme la volonté des

utilisateurs d'accéder à l'information *Action History*. La seconde réponse la plus donnée est d'utiliser l'historique dans le but de déterminer l'auteur et la date de changement et donc de voir l'information *Authorship*.

Améliorations

Les parties suivantes du questionnaire se portent sur l'amélioration de l'historique, en particulier la facilité de lecture des modifications et l'utilité de voir certaines actions apparaitre. Ensuite, 4 idées d'amélioration leur sont proposées et il est demandé d'en évaluer l'intérêt. Enfin, pour la mise en valeur des modifications, une proposition de notation est réalisée. Il est demandé aux étudiants d'interpréter la mise en valeur des modifications proposés.

Facilité de lecture des modifications

Les étudiants ont été sondés sur la facilité de lecture des ajouts, suppressions, renommages, changements de valeur, réagencements et modifications sur plusieurs diagrammes. Le tableau 8.2 résume la part des étudiants interrogés jugeant l'information facile à lire lorsque l'historique est lancé. Cette étude permet de dire que les ajouts sont facilement lisibles tout comme les réagencements contrairement aux modifications. Les ajouts et réagencements sont affichés à l'aide d'une animation. Une première piste d'amélioration pour faciliter la lecture des modifications pourrait être de dire que les animations sont plus intuitives pour les utilisateurs. La lisibilité des autres modifications est donc à améliorer.

Table 8.2 – Facilité de lecture en fonction des différents types de modifications

Type de modification	Facilité de lecture (%)
Ajouts	83
Suppressions	59
Renommages	38
Changements de valeur	31
Réagencements	76
Modifications sur plusieurs diagrammes	34

Proximité des actions

Une nouvelle question a été posée aux étudiants concernant la proximité des actions qu'ils souhaitent voir apparaitre dans l'historique. La

question posée est la suivante : Lorsque vous vous connectez à GenMyModel, aimeriez-vous savoir si des changements ont eu lieu...

— sur un diagramme sur lequel vous avez travaillé

— sur un diagramme sur lequel les autres travaillent mais pas vous

— sur un diagramme impactant un diagramme sur lequel vous travaillez

32 étudiants ont répondu à cette question. 97% veulent savoir si des changements ont eu lieu sur un diagramme sur lequel ils ont travaillé et sur un diagramme impactant un diagramme sur lequel ils ont travaillé. Par contre, ils sont plus mitigés sur des diagrammes sur lesquels ils n'ont pas travaillé puisque 50% des utilisateurs ne souhaitent pas avoir connaissance de modifications dans ce cas là.

Utilité des actions

Une liste d'actions a été donnée aux étudiants. Il devaient déterminer s'il s'agit d'une informations utile, à afficher dans l'historique. Le tableau 8.3 montre les différentes actions à juger ainsi que le pourcentage de réponses jugeant l'information utile. Pour détecter la cohérence des réponses, une question "piège" a été ajoutée à savoir de donner l'inverse de la réponse à la question 1. Cette question a provoqué l'élimination des réponses de 16 étudiants de cette partie du questionnaire.

Table 8.3 – Utilité des différentes actions

Type de modification	Utilité (%)
Création d'un élément	95
Suppression d'un élément	100
Déplacement d'un élément	62
Déplacement d'un élément dans un autre conteneur	71
Redimensionnement d'un élément	24
Changement de place d'un attribut dans une classe	19
Changement de propriété d'une classe	71
Changement de propriété d'un attribut ou d'une opération	71
Changement de cardinalité d'un attribut	67
Changement de type d'un attribut ou d'une opération	67
Changement de nom d'un attribut ou d'une opération	67
Ajout/Suppression d'un paramètre à une opération	76
Changement de type d'une paramètre d'une opération	67
Changements sur une association	71

Les résultats permettent de mettre en évidence que deux informations ne sont pas nécessaires : le redimensionnement des éléments et le changement de place d'un attribut dans une classe. Il est donc possible de ne pas afficher ces informations dans l'historique

Notation de nouvelles fonctionnalités

A ce stade, les étudiants devaient noter l'utilité de quatre nouvelles fonctionnalités sur une échelle de 1 à 4. Les fonctionnalités à noter sont :

— Mise en valeur des éléments de modèle concernés par la modification affichée par l'historique

— Notification incitant à lancer l'historique lorsqu'il y a eu des changements depuis votre déconnexion

— Affichage d'un historique des modifications concernant uniquement des éléments de modèle de votre choix - ou un sous-ensemble de votre choix

— Filtres de recherche sur les auteurs de changements, le type de changement, un intervalle de temps...

32 étudiants ont noté les trois premières fonctionnalités et 31 la dernière.

Table 8.4 – Notation de nouvelles fonctionnalités de l'historique

Fonctionnalité	Moyenne	Médiane
Mise en valeur des éléments	3,2	3
Notification	3	3
Historique localisé	2,8	3
Filtres	2,9	3

Les résultats présents dans le tableau 8.4 montrent l'utilité des quatre fonctionnalités proposées.

Piste pour la mise en valeur d'éléments de modèle

Finalement, il a été demandé aux étudiants ce qu'ils pouvaient comprendre des éléments de modèles mis en valeur de deux manières différentes : par une coloration et par une iconographie.

La figure 8.10 montre les deux façons et le pourcentage de compréhension correcte face à elles. La première solution par coloration ne s'est pas montrée efficace. Plutôt que de comprendre que la coloration rouge

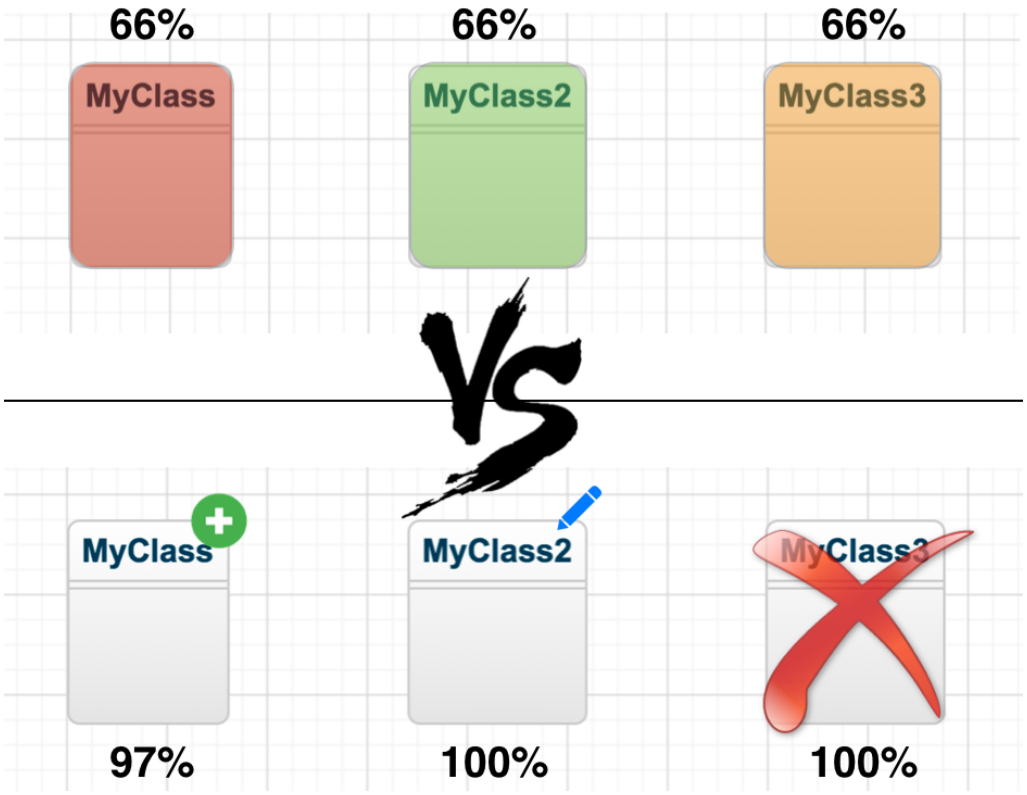


FIGURE 8.10 – Confrontation entre une mise en valeur des éléments de modèle par coloration et par iconographie

se réfère à un élément supprimé, verte à une création et orange à une modification, un grand nombre a interprété la coloration d'une manière différente. Pour eux, la coloration rouge indique des erreurs, la verte une indication qu'aucun problème n'est présent et l'orange alerte sur des possibles erreurs. La mise en valeur par iconographie, elle, ne présente pas de soucis de compréhension.

Usages après plusieurs mois d'utilisation

Les usages de l'historique ont été évalués une seconde fois avec les étudiants dans le but de vérifier si ceux-ci évoluaient au cours du temps. Les mêmes étudiants ont été réévalués ainsi que deux groupes supplémentaires portant le nombre de personnes interrogées à 47 dont 5 femmes et 42 hommes âgés de 21 à 45 ans. Cette seconde évaluation est survenue à la fin de leur projet. Pour la plupart, ils ont utilisé plus longuement

l'outil et donc la fonctionnalité d'historique. Sur les 47 étudiants interrogés, seuls 9 n'ont pas utilisé la fonctionnalité et ont donc été exclus des résultats.

Table 8.5 – Part de réponse des usages de l'historique après plusieurs mois d'utilisation

Usage de l'historique	Part de réponse(%) après quelques heures	Part de réponse(%) après plusieurs mois
Changements réalisés par les autres	80	76
Suppressions	9	11
Ajouts	23	29
Modifications	34	34
Toutes depuis déconnexion	49	45
Autre	6	0
Qui a réalisé les changements et quand	57	66
Voir les modifications depuis déconnexion	40	53
Pouvoir demander des informations	23	16
Autre	3	3
Restaurer une ancienne version	43	53
Annuler les modifications des autres	26	34
Tester une solution	23	32
Autre	0	0
Pourquoi des changements ont été réalisés	17	18
Obligation de l'utiliser	11	0
Autre	14	0

Le tableau 8.5 montre l'évolution des résultats entre les deux analyses. Dans l'ensemble, ceux-ci sont plutôt stables. L'information majeure retenue de ces résultats est la plus forte utilisation de la fonction de restauration de l'historique dans le but d'annuler les modifications réalisées par les autres ou encore de faire des tests de solutions et de revenir à une version antérieure. L'usage à long terme renforce le besoin d'utiliser l'historique.

8.10 Limites de l'étude

Cette étude a été réalisée avec des étudiants de Master 1 ayant déjà une expérience de la modélisation. Néanmoins, leur utilisation pourrait différer de l'usage par des professionnels. Dans le but de consolider cette étude, elle devrait être réalisée une nouvelle fois avec des professionnels. Il est néanmoins compliqué de réalisé ce genre d'études avec des professionnels puisqu'il faudrait interroger des entreprises clientes de GenMyModel. Ces clients payent pour y accéder et il n'est pas toujours très bien vu de les contacter pour réaliser des études.

8.11 Discussions et amélioration de l'historique

Après avoir présenté de manière descriptive les résultats, je vais maintenant les interpréter et ensuite proposer de nouvelles améliorations pour l'historique.

Interprétation des résultats

L'AttrakDiff ainsi que l'étude sur les usages montrent que l'historique remplit pleinement son rôle en étant attractif, facile d'utilisation, stimulant et soutient une fonction sociale. Les étudiants ont utilisé l'historique dans le but d'accéder aux informations d'awareness ciblées. L'étude a également permis de soulever de nombreux points d'améliorations.

Tout d'abord, l'historique est utilisé majoritairement dans le but de suivre les changements réalisés sur le modèle depuis la déconnexion de l'utilisateur. Un système de notification incitant les utilisateurs à lancer l'historique lorsque des modifications ont eu lieu depuis leur déconnexion est jugée comme utile. Ces différents points montrent la nécessité de devoir récapituler les événements manqués lors d'une déconnexion.

Ensuite, peu d'actions sont inutiles lors du déroulement de l'historique. Comme toutes les actions ne sont pas bien visibles, il faut donc trouver une solution pour mettre en valeur les modifications autres que les créations et les réagencements. Pour cela, une première piste a été donnée avec le système d'iconographie plutôt que la coloration des éléments. Cette fonctionnalité de mise en valeur est largement jugée comme étant utile par les étudiants.

Au fur et à mesure que le projet grandit, les utilisateurs ont besoin de restaurer des versions plus anciennes pour annuler des modifications ou pouvoir tester des solutions. Ces restaurations peuvent s'avérer problématiques lors d'un travail collaboratif puisque les changements effectués entre les deux versions sont perdus.

Les résultats de l'étude vont aboutir à quelques améliorations.

Améliorations à apporter

Visibilité des changements

L'amélioration de la visibilité des changements permettrait aux utilisateurs d'identifier plus rapidement les changements effectués à chaque étape de l'historique. Cette visibilité serait surtout nécessaire lorsque le changement du point de vue graphique est minime. Par exemple, dans le cas d'un changement de cardinalité sur une association, la perception

du changement est infime. Il est courant d'utiliser des variables visuelles comme celles définies par Bertin[Bertin, 2010] et appliquées par Moody à la modélisation[Moody, 2009]. Les variables visuelles sont au nombre de 8 : la position horizontale, la position verticale, la forme, la taille, la couleur, la luminosité, l'orientation et la texture. Notre étude a montré que les icônes étaient plus compréhensibles que la coloration. De ce fait, la variable visuelle de forme doit être utilisée. La complexité de la mise en place de cette nouvelle fonctionnalité résidera dans le fait de trouver une iconographie compréhensible par les utilisateurs et sans ambiguïté pour un grand nombre d'actions possibles.

Remise en contexte de l'utilisateur

Lors de ma première étude sur l'analyse des sessions d'activité des utilisateurs, j'avais remarqué que le temps passé entre la connexion et la première action était important. A ce moment, je n'y ai pas attaché une grande importance mais en réalité, il s'agit du temps de remise en contexte de l'utilisateur, remis en avant lors de la validation de l'historique. Cette remise en contexte est donc importante pour l'utilisateur et il est nécessaire de l'améliorer. Pour cela, lorsque l'utilisateur se connecte à son projet et que des modifications ont été réalisées depuis sa dernière déconnexion, l'historique pourrait lui être proposé en le positionnant directement au moment de sa dernière déconnexion. Ainsi, l'utilisateur peut voir rapidement toutes les modifications réalisées.

Filtrer les commandes

Lors de la validation de l'historique sur les usages, j'ai mis en avant les scénarios d'utilisation de l'historique. L'historique sert majoritairement à retrouver des commandes précises ou l'auteur de commandes. Pour aider l'utilisateur à trouver cette information plus rapidement, des filtres de recherche pourraient être mis en place. Grâce à ces filtres, l'utilisateur pourrait renseigner un intervalle de temps, un auteur ou un type de commandes et le système afficherait à l'utilisateur toutes les commandes satisfaisant les conditions. Il pourrait alors accéder plus rapidement à l'information qu'il recherche.

Modifier le passé

L'un des usages de l'historique est de revenir à un état antérieur sur le modèle. Cette action n'est pas sans impact. Il se peut que d'autres actions importantes aient eu lieu entre la version récupérée et la version

antérieure. Une solution envisageable pour palier ce problème serait de pouvoir récupérer des actions passées pour les rétablir dans le modèle courant ce qui implique de modifier le passé pour récupérer des changements dans le présent. La mise en place de cette fonctionnalité n'est pas triviale. En effet, il faut pouvoir déterminer la profondeur des modifications à récupérer. Par exemple, si je souhaite récupérer une classe et que celle-ci est liée à d'autres classes qui n'existent pas dans le modèle actuel, doit-on récupérer uniquement la classe ou la classe avec toutes les dépendances ? En plus de ces problématiques, il faut pouvoir mesurer l'impact des changements réalisés dans le passé. Des travaux existent déjà sur le fait de les mesurer dans le présent[Briand et al., 2003] en analysant les effets de bord d'une modification sur le modèle. Cette analyse repose sur des règles de calcul de distance pour déterminer les éléments de modèle impactés par des modifications. Néanmoins, modifier le passé est plus complexe puisqu'il faut prendre en compte les étapes intermédiaires entre la version passée modifiée et la version présenté du modèle. Ceci ouvre des perspectives de mesure plus riches et donc plus pertinentes pour les concepteurs. Nous espérons vraiment pouvoir explorer cette piste dans un futur proche.

Bilan sur les fonctionnalités collaboratives

Les différentes fonctionnalités implémentées permettent d'afficher les informations d'awareness les plus importantes définies lors de notre étude comme le montre le tableau 8.6.

Table 8.6 – Support de l'awareness dans GenMyModel

	Fonctionnalités			
	List des utilisateurs	Focus + Gestion des tâches	Changements en direct	Historique
Presence	✓			
Identity	✓			
Authorship	✓	✓		
Action			❍	
Action History				✓
Intention		✓		
Artifact				✓
Artifact History				✓

✓ : information fournie
❍ : information fournie partiellement

Toutes les études autour de l'historique ont permis de montrer les différents usages de la fonctionnalité mais surtout de dire que l'implémentation correspond à la pratique de la conception logicielle. Ce tableau montre principalement qu'il reste une information d'awareness à

supporter pleinement : *Action*. Le problème majeur est de pouvoir montrer toutes les actions ayant lieu. La solution actuelle ne permet pas de voir les actions hors de la zone d'affichage de l'utilisateur. Pour palier ce problème, plusieurs solutions sont envisageables.

— Une liste des modifications affiche de manière textuelle toutes les modifications en cours sur le modèle. Chaque modification devra afficher de manière succincte une description ainsi que le(s) diagramme(s) impacté(s).

— Colorer le nom des diagrammes de la couleur d'un utilisateur lorsque celui-ci effectue une modification. Ainsi, les personnes travaillant sur un autre diagramme peuvent savoir qu'une modification est en cours. Néanmoins, cette solution ne permet pas de montrer le contenu de la modification à l'utilisateur directement. De plus, si la modification a lieu dans le même diagramme mais hors de la zone d'affichage, l'utilisateur ne la verra tout de même pas.

Résumé de la partie

La seconde partie de cette thèse a mis en avant le support incomplet des informations d'awareness les plus importantes, définies par mon étude. De nouvelles fonctionnalités ont du être conçues pour fournir un support total de ces informations dont l'historique, sur lequel plusieurs validations ont eu lieu. En effet, la représentation de l'historique est totalement nouvelle, il était donc nécessaire de vérifier les usages et son adoption par les utilisateurs.

Conclusion et perspectives

Cette thèse s'inscrit dans le cadre du développement distribué, en particulier la modélisation, durant lequel plusieurs personnes vont travailler ensemble dans le but de mener à bien un projet.

Ma thèse s'est déroulée dans un contexte industriel dont l'objectif était de mettre en place la collaboration au sein des outils de modélisation. La première partie de ma thèse a montré les 5 phases à supporter pour rendre un outil de modélisation collaboratif : La récupération du modèle, son stockage, la représentation des changements, la communication des changements ainsi que la gestion des conflits. Cette partie s'est ensuite portée sur un système d'échange de messages pour que chaque action d'un utilisateur soit immédiatement répercutée chez les autres utilisateurs connectés au modèle. Le système de commandes dAMOCleS est générique et applicable à tous les outils de modélisation. Il permet de mettre en place rapidement l'échange de messages au sein de l'outil grâce à sa généricité. Une fois le support de ces 5 phases réalisé, mes travaux se sont ensuite portés sur l'assistance aux utilisateurs pour leur faciliter la collaboration avec comme notion centrale l'awareness. Cette notion recense les informations dont les utilisateurs ont besoin à propos de l'activité sur l'environnement partagé, les tâches et les collaborateurs pour pouvoir réaliser leur propre activité sans empiéter sur celle des autres. Dans le but d'être au plus proche du quotidien des concepteurs, des études ont été menées avec eux afin de déterminer les différents modes de collaboration pour ensuite classer pour chacun les informations d'awareness les plus importantes.

De cette étude, différentes fonctionnalités collaboratives ont été implémentées pour répondre à ces besoins, notamment la gestion des droits, un panneau de discussions, une gestion des tâches, l'affichage des focus des collaborateurs, la liste des collaborateurs connectés, un gestionnaire de tâches et enfin un historique des modifications. Ce dernier a nécessité une étude plus approfondie puisque cette fonctionnalité est très peu présente dans les outils de modélisation. Notre proposition s'avère innovante dans la manière de la représenter : sous forme d'une vidéo.

La validation de l'historique s'est déroulée en deux temps. D'abord une étude exploratoire a été réalisée de manière à vérifier que les usages correspondaient aux buts initiaux. Ensuite, une étude approfondie a permis de déterminer précisément les usages, les améliorations possibles et l'attractivité envers la fonctionnalité. Cette validation a permis de montrer que la solution correspond aux usages des concepteurs mais aussi à

leurs besoins.
Ce type de validation n'est pas courant. Pourtant celle-ci permet de tirer énormément de leçons comme les points d'amélioration pour que les utilisateurs puissent adopter une fonctionnalité. La seule contrainte est la nécessité de disposer d'un panel d'utilisateurs finaux. Le contexte industriel de ma thèse m'ayant permis d'être au contact de ceux-ci, cette barrière fut levée et l'intégralité de ma thèse a pu se faire en apportant des solutions au plus proche de leurs besoins.

La réalisation de ces travaux a permis d'ouvrir de nouvelles perspectives. Tout d'abord, la collaboration de manière distribuée implique des problèmes de communication, issus des différences de langage ou de culture, non résolus à l'heure actuelle. Les jeux vidéos partagent les mêmes problématiques et certains jeux, notamment Portal 2 [1], proposent un langage universel, à base d'icônes pour pouvoir interagir avec les joueurs connectés. Dans ce jeu, les personnes "posent" des icônes dans le jeu pour pouvoir donner des directives à leurs coéquipiers. En s'inspirant de ce principe, il serait possible de définir un langage permettant aux utilisateurs de communiquer dans l'outil de modélisation au travers d'un système d'icônes. Pour ce faire, les utilisateurs pourraient ajouter ces icônes sur le modèle, un élément de modèle ou un ensemble d'éléments pour pouvoir définir des tâches aux autres membres de l'équipe. Le système ne donnerait que des actions possibles, paramétrées par leur localisation. Différentes études seraient nécessaires pour adapter ce principe à la modélisation. Tout d'abord, il faudrait définir l'ensemble des actions possibles comme par exemple proposer un réagencement des éléments de modèle, ou encore proposer de changer le nom de certains éléments pour qu'il soit plus explicite, etc. Suite à la définition de l'ensemble des actions, il faudrait définir les icônes qualifiant celles-ci. Pour ce faire, il sera nécessaire de réaliser une étude durant laquelle plusieurs icônes seront proposées et les personnes de nationalité et/ou culture différentes répondant à l'étude devront définir le sens de chacune. Une fois les actions définies ainsi que les icônes associées, il faudra se poser plusieurs questions. Doit-on indiquer la progression des différentes tâches ? Un historique des tâches doit-il apparaitre sur chaque élément de modèle ? Comment cela se passe t-il dans le cas où la tâche concerne plusieurs éléments de modèle ? Et si la tâche s'applique sur plusieurs diagrammes ? Est-ce que l'affichage de ces tâches doit être permanent ?

D'ailleurs, il serait intéressant de quantifier le gain de temps, les risques d'erreur et le confort entre un système d'icônes et du texte (toujours lo-

1. http ://www.thinkwithportals.com/

calisé sur les éléments concernés). Pour l'affichage permanent ou non des tâches, plusieurs pistes sont envisageables. Premièrement, il serait possible de laisser cet affichage permanent. Ensuite, il serait possible de donner une priorité à ces tâches pour ensuite décider d'une stratégie d'affichage. Par exemple, si une tâche a été définie comme critique, l'icône associée à la tâche pourrait clignoter. Dans le cas d'une tâche non prioritaire, un rappel pourrait s'afficher périodiquement pour notifier à l'utilisateur de ne pas l'oublier. Ce système d'icônes pourrait également être étendu à d'autres problématiques comme l'affichage de l'information d'awareness *Opinion*. Un utilisateur pourrait demander l'avis des autres membres de l'équipe et ceux-ci pourraient lui répondre par une icône correspondant à leur avis (à l'instar de Facebook où à la place d'aimer une publication, les personnes peuvent nuancer leurs avis avec un *smiley* [2]).

Enfin, il serait intéressant d'étudier comment ces perspectives présentées ici pour les outils de modélisation pourraient être appliquées aux autres étapes du cycle de production logicielle (analyse des besoins, programmation,etc.).

Une autre piste amenant de nouveaux travaux de recherche a été soulevée durant la validation de l'historique des modifications : pouvoir modifier le passé. Au fur et à mesure de l'avancée d'un projet, les utilisateurs utilisent plus fréquemment l'historique pour annuler des modifications ou revenir à un état antérieur suite à des erreurs. Ces restaurations impliquent des pertes d'information. Pour palier ces pertes, le fait de modifier directement le modèle dans le passé, ou de récupérer certaines actions antérieures, permettrait de les limiter. Comme dit précédemment, modifier le passé n'est pas simple puisqu'il faut définir le ou les éléments à récupérer ainsi qu'évaluer les impacts sur le modèle courant. Les travaux existants sur l'analyse des impacts ne peuvent pas s'appliquer de la même manière pour déterminer les impacts de modifications dans le passé. En effet, entre la version du modèle dans le passé sur lequel l'utilisateur souhaite récupérer des modifications et le modèle courant, d'autres modifications ont eu lieu entre temps. Il ne faut donc plus raisonner sur un seul modèle mais sur deux avec un ensemble de modifications dont l'analyse d'impacts ne tient pas compte. La première chose à faire avant de démarrer ce travail est de prouver l'intérêt de cette approche : À l'aide de 2 ou 3 cas d'usage réalistes (c'est-à-dire avec des diagrammes UML fournis et des modifications que l'on retrouve dans la pratique quotidienne des développeurs), il faudra évaluer l'apport de

2. http ://newsroom.fb.com/news/2016/02/reactions-now-available-globally/

notre nouvelle approche pour ce qui est de la mesure d'impact par rapport à une approche statique habituelle, c'est-à-dire simplement sur la structure. En se basant sur le système de commandes défini dans dAMOCleS, le suivi des changements entre deux versions d'un modèle est simplifié. Il est ainsi possible de déterminer l'ensemble des modifications liées à des éléments de modèle pour définir ceux qui doivent être récupérés. Les travaux à mener se feront donc en deux phases. La première consistera à établir les règles de récupération des éléments de modèles à partir des actions de l'utilisateur. Ces actions devront également être préalablement listées. Ensuite, en s'inspirant des travaux sur l'analyse des impacts, l'outil devra être en mesure de notifier à l'utilisateur tous les problèmes pouvant survenir lors de la récupération des éléments de modèle. dAMOCleS ayant été conçu dans le but d'être applicable pour tous les outils de modélisation, ces travaux se feront dans le même cadre. Ce travail renvoie à l'opposition historique entre sauvegarde des actions ou sauvegarde de la structure. Au vu de l'activité importante de la communauté scientifique sur l'analyse de code, des corrections automatiques de bug, etc, l'approche "par actions" pourrait être une alternative pertinente à de nombreux problèmes. L'étude possible présentée ici dans le cadre de la modélisation logicielle pourrait être donc une contribution à l'ensemble de la communauté logicielle.

That's all folks !

Liste des Publications

Articles

1. **Software Support Requirements for Awareness in Collaborative Modeling**. Michel Dirix, Xavier Le Pallec, Alexis Muller. *CoopIS'14 (Rang A)*
2. **Awareness in Computer-Supported Collaborative Modelling. Application to GenMyModel**. Michel Dirix. *Doctoral Symposium - ECOOP'13*

Poster

1. **GenMyModel : An Online UML Case Tool**. Michel Dirix, Alexis Muller, Vincent Aranega. *ECOOP'13*

A Questionnaire AttrakDiff

Intérêt de ce questionnaire

Comme vous le savez peut-être, vos enseignants sont impliqués dans des activités de recherche. Ce questionnaire va servir à faire avancer la recherche dans le domaine de la modélisation UML.
L'historique que vous avez utilisé au sein de GenMyModel est une nouveauté. Pour cela, nous devons connaitre au travers de ce questionnaire quel est votre ressenti face à cet historique et connaitre également la façon dont vous l'utilisez. C'est pourquoi vos réponses ne doivent pas être données au hasard et doivent être pertinentes.

Questions générales

Ces réponses seront traitées de manière anonyme et ne seront exploitées que pour réaliser des statistiques.

Votre âge :

Votre sexe :

Votre niveau d'expertise en modélisation sur une échelle de 1 à 4 (1 étant débutant et 4 expert):

1 ◯ ◯ ◯ ◯ 4

Votre niveau d'étude actuel:

- ❏ Licence 3
- ❏ Master 1
- ❏ Master 2

Le nombre de personnes dans votre équipe (y compris vous) :

Le nombre d'éléments de modèles dans votre projet*(voir page suivante pour savoir comment récupérer ces informations) :

Element:
Graphical elements:

Vous avez des remarques à faire, des suggestions ou des idées? C'est le moment!

*Pour récupérer le nombre d'éléments dans votre projet, ouvrez le catsoo shell situé en bas à gauche de la page GenMyModel. Rendez-vous sur http://michel-dirix.com/script.html et copiez-collez "log('Elements = ' + (__ROOT allSubObjects()[Element] size() - 2)); log('Graphical Elements = ' + graphicElements() size());" dans le catsoo shell. Ecrivez ensuite les deux valeurs dans le champs de réponse de ce questionnaire.

Généralités sur GenMyModel

Avez-vous rencontré des problèmes avec l'outil GenMyModel?

Avez-vous des idées de fonctionnalités?

Votre ressenti face à l'historique

Ce questionnaire va maintenant se focaliser sur l'historique. L'historique n'a jamais été présenté sous cette forme dans les outils de modélisation. Nous devons donc tester l'utilisabilité de l'historique et votre ressenti.
Pour cela, la première partie de questionnaire se présente sous forme de paires de mots pour vous assister dans l'évaluation du système. Chaque paire représente des contrastes. Les échelons entre les deux extrémités vous permettent de décrire l'intensité de la qualité choisie. Ne pensez pas trop aux paires de mots et essayez simplement de **donner une réponse spontanée**. Vous pourrez avoir l'impression que certains termes ne décrivent pas correctement le système. Dans ce cas, assurez vous de donner tout de même une réponse. Gardez à l'esprit qu'il n'y a pas de bonne ou mauvaise réponse. **Seule votre opinion compte !** Vos réponses seront enregistrées et traitées de manière anonyme.
N'oubliez pas que vos réponses concernent uniquement l'historique et non l'outil en général.

Humain	○○○○○○○	Technique
M'isole	○○○○○○○	Me sociabilise
Plaisant	○○○○○○○	Déplaisant
Original	○○○○○○○	Conventionnel
Simple	○○○○○○○	Compliqué
Professionnel	○○○○○○○	Amateur
Laid	○○○○○○○	Beau
Pratique	○○○○○○○	Pas pratique
Agréable	○○○○○○○	Désagréable
Fastidieux	○○○○○○○	Efficace
De bon goût	○○○○○○○	De mauvais goût
Prévisible	○○○○○○○	Imprévisible
Bas de gamme	○○○○○○○	Haut de gamme
M'exclut	○○○○○○○	M'intègre
Me rapproche des autres	○○○○○○○	Me sépare des autres
Non présentable	○○○○○○○	Présentable
Rebutant	○○○○○○○	Attirant
Sans imagination	○○○○○○○	Créatif
Bon	○○○○○○○	Mauvais
Confus	○○○○○○○	Clair
Repoussant	○○○○○○○	Attrayant

Audacieux	○○○○○○○	Prudent
Novateur	○○○○○○○	Conservateur
Ennuyeux	○○○○○○○	Captivant
Peu exigeant	○○○○○○○	Challenging
Motivant	○○○○○○○	Décourageant
Nouveau	○○○○○○○	Commun
Incontrôlable	○○○○○○○	Maîtrisable

Votre utilisation de l'historique

Quelle(s) information(s) recherchiez-vous lorsque vous avez utilisé l'historique?

- ❑ Les changements réalisés par les autres
  - en particulier:
  - ❑ les suppressions
  - ❑ les ajouts
  - ❑ les modifications
  - ❑ tous les changements ayant eu lieu après ma déconnexion
  - ❑ autre:
- ❑ Qui a réalisé des changements et quand
  - en particulier:
  - ❑ voir s'il y a eu des modifications après ma déconnexion
  - ❑ pouvoir demander des informations à la personne concernée
  - ❑ autre
- ❑ Restaurer une ancienne version
  - en particulier:
  - ❑ annuler des modifications faites par d'autres
  - ❑ j'ai testé une solution et je reviens en arrière
  - ❑ autre:
- ❑ Pourquoi des changements ont été réalisés
- ❑ Je ne cherchais rien, j'ai été obligé de l'utiliser
- ❑ Autre raison :

Pourquoi cherchiez-vous cette information?

Lors de l'utilisation de l'historique, à quand remontait votre dernière connexion à l'outil?

- ❏ quelques heures
- ❏ la veille
- ❏ plusieurs jours
- ❏ plusieurs semaines

Voyez-vous des améliorations à apporter à l'historique?

Lors de la lecture de l'historique, voyez-vous facilement…

- les ajouts oui - non
- les suppressions oui - non
- les renommages oui - non
- les changements de valeurs oui - non
- les réagencements oui - non
- les modifications sur plusieurs diagrammes oui - non

Lorsque vous vous connectez à GenMyModel, aimeriez-vous savoir si des changements ont eu lieu…

- sur un diagramme sur lequel vous avez travaillé oui - non
- sur un diagramme sur lequel les autres travaillent mais pas vous oui - non
- sur un diagramme impactant un diagramme sur lequel vous travaillez oui - non

Voici une liste d'actions possibles, cochez selon si vous jugez utile ou non de voir cette action dans l'historique:

action	utile	inutile
création d'un élément (un élément désigne tous les éléments pouvant être créés à partir de la palette de création dans GenMyModel)		
suppression d'un élément		
déplacement d'un élément		
déplacement d'un élément dans un autre conteneur (par ex. déplacement d'une classe dans un package)		

redimensionnement d'un élément		
changement de place d'un attribut dans une classe		
changement de propriété d'une classe (abstraite, publique, privée)		
changement de propriété d'un attribut ou d'une opération (statique, public, privé)		
changement de cardinalité d'un attribut		
l'inverse de la réponse de la ligne une de ce tableau		
changement de type d'un attribut ou d'une opération		
changement de nom d'un attribut ou d'une opération		
ajout/suppression d'un paramètre à une opération		
changement de type d'un paramètre d'une opération		
changements sur une association (cardinalité, noms…)		

Les améliorations de l'historique

Noter les améliorations suivantes sur une échelle de 1 à 4. 1 signifie que vous trouvez l'amélioration inutile et 4 signifie que l'amélioration serait nécessaire.

- Mise en valeur des éléments de modèle concernés par la modification affichée par l'historique

1 ○ ○ ○ ○ 4

- Notification incitant à lancer l'historique lorsqu'il y a eu des changements depuis votre déconnexion

1 ○ ○ ○ ○ 4

- Affichage d'un historique des modifications concernant uniquement des éléments de modèle de votre choix - ou un sous-ensemble de votre choix.

1 ○ ○ ○ ○ 4

- Filtres de recherche sur les auteurs de changements, le type de changement, un intervalle de temps…

1 ○ ○ ○ ○ 4

Si en ouvrant l'historique, vous voyez des classes colorées de cette façon, que pouvez-vous en déduire?

Réponse:

Et décorées de cette façon, que pouvez-vous en déduire?

Réponse:

Merci pour vos réponses et votre soutien à la recherche! ;)

Bibliographie

[Axellience, 2015] Axellience (2015). Genmymodel. https ://www.genmymodel.com/.

[Bertin, 2010] Bertin, J. (2010). *Semiology of Graphics - Diagrams, Networks, Maps*. ESRI.

[Biehl et al., 2007] Biehl, J. T., Czerwinski, M., Smith, G., and Robertson, G. G. (2007). Fastdash : a visual dashboard for fostering awareness in software teams. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1313–1322. ACM.

[Blanc et al., 2008] Blanc, X., Mounier, I., Mougenot, A., and Mens, T. (2008). Detecting model inconsistency through operation-based model construction. In *Software Engineering, 2008. ICSE'08. ACM/IEEE 30th International Conference on*, pages 511–520. IEEE.

[Bolt, 1990] Bolt, R. A. (1990). A gaze-responsive self-disclosing display. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 3–10. ACM.

[Bouras et al., 2003] Bouras, C., Giannaka, E., and Tsiatsos, T. (2003). Virtual collaboration spaces : the eve community. In *Applications and the Internet, 2003. Proceedings. 2003 Symposium on*, pages 48–55. IEEE.

[Briand et al., 2003] Briand, L. C., Labiche, Y., and Sullivan, L. (2003). Impact analysis and change management of uml models. In *Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference on*, pages 256–265. IEEE.

[Brown, 2004] Brown, A. W. (2004). Model driven architecture : Principles and practice. *Software and Systems Modeling*, 3(4) :314–327.

[Carstensen and Schmidt, 1999] Carstensen, P. H. and Schmidt, K. (1999). Computer supported cooperative work : New challenges to systems design. In *In K. Itoh (Ed.), Handbook of Human Factors*. Citeseer.

[Chevalier et al., 2010] Chevalier, F., Dragicevic, P., Bezerianos, A., and Fekete, J.-D. (2010). Using text animated transitions to support navigation in document histories. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 683–692. ACM.

[Cinergix, 2015] Cinergix (2015). Creately. http ://creately.com/.

[Claypool et al., 2001] Claypool, M., Brown, D., Le, P., and Waseda, M. (2001). Inferring user interest. *Internet Computing, IEEE*, 5(6) :32–39.

[Cramton, 2001] Cramton, C. D. (2001). The mutual knowledge problem and its consequences for dispersed collaboration. *Organization science*, 12(3) :346–371.

[Damian et al., 2007] Damian, D., Izquierdo, L., Singer, J., and Kwan, I. (2007). Awareness in the wild : Why communication breakdowns occur. In *Global Software Engineering, 2007. ICGSE 2007. Second IEEE International Conference on*, pages 81–90. IEEE.

[DeFranco-Tommarello and Deek, 2002] DeFranco-Tommarello, J. and Deek, F. P. (2002). Collaborative software development : a discussion of problem solving models and groupware technologies. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 568–577. IEEE.

[Dirix et al.,] Dirix, M., Muller, A., and Aranega, V. Genmymodel : An online uml case tool. *Joint Proceedings of Tools, Demos & Posters*, page 14.

[Dourish and Bellotti, 1992] Dourish, P. and Bellotti, V. (1992). Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, CSCW '92, pages 107–114, New York, NY, USA.

[Ellis et al., 1991] Ellis, C. A., Gibbs, S. J., and Rein, G. (1991). Groupware : some issues and experiences. *Communications of the ACM*, 34(1) :39–58.

[Farwick et al., 2010] Farwick, M., Agreiter, B., White, J., Forster, S., Lanzanasto, N., and Breu, R. (2010). A web-based collaborative metamodeling environment with secure remote model access. In *Web Engineering*, volume 6189 of *Lecture Notes in Computer Science*, pages 278–291.

[Fette and Melnikov, 2011] Fette, I. and Melnikov, A. (2011). The websocket protocol.

[Fuks et al., 2005] Fuks, H., Raposo, A. B., Gerosa, M. A., and Lucena, C. J. (2005). Applying the 3c model to groupware development. *International Journal of Cooperative Information Systems*, 14(02n03) :299–328.

[Garrett et al., 2005] Garrett, J. J. et al. (2005). Ajax : A new approach to web applications.

[Genilloud and Wegmann, 2000] Genilloud, G. and Wegmann, A. (2000). A foundation for the concept of role in object modelling. In *Enterprise Distributed Object Computing Conference, 2000. EDOC 2000. Proceedings. Fourth International*, pages 76–85. IEEE.

[Gerosa et al., 2006] Gerosa, M. A., Pimentel, M., Fuks, H., and de Lucena, C. J. P. (2006). Development of groupware based on the 3c collaboration model and component technology. In *Groupware : Design, Implementation, and Use*, pages 302–309. Springer.

[Geyer et al., 2001] Geyer, W., Richter, H., Fuchs, L., Frauenhofer, T., Daijavad, S., and Poltrock, S. (2001). A team collaboration space supporting capture and access of virtual meetings. In *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*, pages 188–196. ACM.

[Gross, 1997] Gross, T. (1997). Towards flexible support for cooperation : group awareness in shared workspaces. In *Database and Expert Systems Applications, 1997. Proceedings., Eighth International Workshop on*, pages 406–411. IEEE.

[Gross et al., 2005] Gross, T., Stary, C., and Totter, A. (2005). User-centered awareness in computer-supported cooperative work-systems : Structured embedding of findings from social sciences. *International Journal of Human-Computer Interaction*, 18(3) :323–360.

[Gutwin and Greenberg, 2002] Gutwin, C. and Greenberg, S. (2002). A descriptive framework of workspace awareness for real-time groupware. *Computer Supported Cooperative Work (CSCW)*, 11(3-4) :411–446.

[Gutwin et al., 1996a] Gutwin, C., Greenberg, S., and Roseman, M. (1996a). Workspace awareness in real-time distributed groupware : Framework, widgets, and evaluation. In *Proceedings of HCI on People and Computers XI*, HCI '96, pages 281–298, London, UK, UK.

[Gutwin et al., 1996b] Gutwin, C., Roseman, M., and Greenberg, S. (1996b). A usability study of awareness widgets in a shared workspace groupware system. In *Proceedings of the 1996 ACM conference on Computer supported cooperative work*, pages 258–267. ACM.

[Gutwin et al., 2005] Gutwin, C., Schneider, K., Paquette, D., and Penner, R. (2005). Supporting group awareness in distributed software development. In *Engineering Human Computer Interaction and Interactive Systems*, pages 383–397. Springer.

[Gutwin et al., 1995] Gutwin, C., Stark, G., and Greenberg, S. (1995). Support for workspace awareness in educational groupware. In *The first international conference on Computer support for collaborative learning*, CSCL '95, pages 147–156.

[Hassenzahl, 2005] Hassenzahl, M. (2005). The thing and i : Understanding the relationship between user and product. In Blythe, M., Overbeeke, K., Monk, A., and Wright, P., editors, *Funology*, volume 3 of *Human-Computer Interaction Series*, pages 31–42. Springer Netherlands.

[Hassenzahl, 2008] Hassenzahl, M. (2008). User experience (ux) : towards an experiential perspective on product quality. In *Proceedings of the 20th International Conference of the Association Francophone d'Interaction Homme-Machine*, pages 11–15. ACM.

[Hassenzahl et al., 2003] Hassenzahl, M., Burmester, M., and Koller, F. (2003). Attrakdiff : Ein fragebogen zur messung wahrgenommener hedonischer und pragmatischer qualitat. In Szwillus, G. and Ziegler, J., editors, *Mensch und Computer 2003*, volume 57 of *Berichte des German Chapter of the ACM*, pages 187–196.

[Hattori, 2010] Hattori, L. (2010). Enhancing collaboration of multi-developer projects with synchronous changes. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*, pages 377–380. ACM.

[Hattori and Lanza, 2009a] Hattori, L. and Lanza, M. (2009a). An environment for synchronous software development. In *Software Engineering-Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*, pages 223–226. IEEE.

[Hattori and Lanza, 2009b] Hattori, L. and Lanza, M. (2009b). Mining the history of synchronous changes to refine code ownership. In *Mining Software Repositories, 2009. MSR'09. 6th IEEE International Working Conference on*, pages 141–150. IEEE.

[Hattori and Lanza, 2010] Hattori, L. and Lanza, M. (2010). Syde : a tool for collaborative software development. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*, pages 235–238. ACM.

[Hattori et al., 2012] Hattori, L. P., Lanza, M., and Robbes, R. (2012). Refining code ownership with synchronous changes. *Empirical Software Engineering*, 17(4-5) :467–499.

[Heinrich et al., 2013] Heinrich, M., Grüneberger, F. J., Springer, T., Hauer, P., and Gaedke, M. (2013). Gawi : a comprehensive workspace awareness library for collaborative web applications. In *Web Engineering*, pages 482–485. Springer.

[Herbsleb and Grinter, 1999] Herbsleb, J. D. and Grinter, R. E. (1999). Architectures, coordination, and distance : Conway's law and beyond. *IEEE software*, (5) :63–70.

[Herrmann et al., 2013] Herrmann, T., Nolte, A., and Prilla, M. (2013). Awareness support for combining individual and collaborative process design in co-located meetings. *Computer Supported Cooperative Work (CSCW)*, 22(2-3) :241–270.

[Hill and Gutwin, 2004] Hill, J. and Gutwin, C. (2004). The maui toolkit : Groupware widgets for group awareness. *Computer Supported Cooperative Work (CSCW)*, 13(5-6) :539–571.

[Hossain et al., 2011] Hossain, E., Bannerman, P. L., and Jeffery, D. R. (2011). Scrum practices in global software development : A research framework. In *PROFES*, pages 88–102. Springer.

[Hurwitz and Mallery, 1995] Hurwitz, R. and Mallery, J. C. (1995). The open meeting : A web-based system for conferencing and collaboration. In *Proceedings of the Fourth International Conference on The World-Wide Web*.

[Ivček and Grbac, 2008] Ivček, M. and Grbac, T. G. (2008). Aspects of quality assurance in global software development organization. In *International Convention MIPRO 2008*. Hrvatska znanstvena bibliografija i MZOS-Svibor.

[Jiménez et al., 2009] Jiménez, M., Piattini, M., and Vizcaíno, A. (2009). Challenges and improvements in distributed software development : a systematic review. *Adv. Soft. Eng.*, 2009 :3 :1–3 :16.

[Julian, 2015] Julian, D. P. (2015). Systems and methods for counteracting a perceptual fading of a movable indicator. US Patent 8,937,591.

[Just and Carpenter, 1976] Just, M. A. and Carpenter, P. A. (1976). The role of eye-fixation research in cognitive psychology. *Behavior Research Methods & Instrumentation*, 8(2) :139–143.

[Kahneman, 1973] Kahneman, D. (1973). *Attention and effort*. Citeseer.

[Kirsch-Pinheiro et al., 2003] Kirsch-Pinheiro, M., De Lima, J. V., and Borges, M. R. (2003). A framework for awareness support in groupware systems. *Computers in Industry*, 52(1) :47–57.

[Kirsh, 2000] Kirsh, D. (2000). A few thoughts on cognitive overload.

[Kreifelts et al., 1993] Kreifelts, T., Hinrichs, E., and Woetzel, G. (1993). Sharing to-do lists with a distributed task manager. In *Proceedings of the Third European Conference on Computer-Supported Cooperative Work 13–17 September 1993, Milan, Italy ECSCW'93*, pages 31–46. Springer.

[Kujala et al., 2011] Kujala, S., Roto, V., Väänänen-Vainio-Mattila, K., Karapanos, E., and Sinnelä, A. (2011). Ux curve : A method for evaluating long-term user experience. *Interacting with Computers*, 23(5) :473–483.

[Kuryazov and Winter, 2014] Kuryazov, D. and Winter, A. (2014). Representing model differences by delta operations. In *Enterprise Distributed Object Computing Conference Workshops and Demonstrations (EDOCW), 2014 IEEE 18th International*, pages 211–220. IEEE.

[Lallemand et al., 2015] Lallemand, C., Koenig, V., Gronier, G., and Martin, R. (2015). Création et validation d'une version française du questionnaire attrakdiff pour l'évaluation de l'expérience utilisateur des systèmes interactifs. *Revue Européenne de Psychologie Appliquée/European Review of Applied Psychology*.

[Lamport, 1978] Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7) :558–565.

[Lanza et al., 2013] Lanza, M., D'Ambros, M., Bacchelli, A., Hattori, L., and Rigotti, F. (2013). Manhattan : Supporting real-time visual team activity awareness. In *Program Comprehension (ICPC), 2013 IEEE 21st International Conference on*, pages 207–210. IEEE.

[Lauwers and Lantz, 1990] Lauwers, J. C. and Lantz, K. A. (1990). Collaboration awareness in support of collaboration transparency : requirements for the next generation of shared window systems. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 303–311. ACM.

[Loreto et al., 2011] Loreto, S., Saint-Andre, P., Salsano, S., and Wilkins, G. (2011). Known issues and best practices for the use of long polling and streaming in bidirectional http. *Internet Engineering Task Force, Request for Comments*, 6202(2070-1721) :32.

[Maitland et al., 2006] Maitland, J., Sherwood, S., Barkhuus, L., Anderson, I., Hall, M., Brown, B., Chalmers, M., and Muller, H. (2006). Increasing the awareness of daily activity levels with pervasive compu-

ting. In *Pervasive Health Conference and Workshops, 2006*, pages 1–9. IEEE.

[Mayer and Moreno, 2003] Mayer, R. E. and Moreno, R. (2003). Nine ways to reduce cognitive load in multimedia learning. *Educational psychologist*, 38(1) :43–52.

[Mens, 2002] Mens, T. (2002). A state-of-the-art survey on software merging. *Software Engineering, IEEE Transactions on*, 28(5) :449–462.

[Moody, 2009] Moody, D. (2009). The physics of notations : Toward a scientific basis for constructing visual notations in software engineering. *Software Engineering, IEEE Transactions on*, 35(6) :756–779.

[Mougenot et al., 2009] Mougenot, A., Blanc, X., and Gervais, M.-P. (2009). D-praxis : A peer-to-peer collaborative model editing framework. In *Distributed Applications and Interoperable Systems*, pages 16–29. Springer.

[Nulab, 2015] Nulab (2015). Cacoo. https ://cacoo.com.

[OMG, 2015] OMG (2015). The xmi specifications. http ://www.omg.org/spec/XMI/.

[Omoronyia et al., 2010] Omoronyia, I., Ferguson, J., Roper, M., and Wood, M. (2010). A review of awareness in distributed collaborative software engineering. *Software : Practice and Experience*, 40(12) :1107–1133.

[Posner and Baecker, 1992] Posner, I. R. and Baecker, R. M. (1992). How people write together [groupware]. In *System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on*, volume 4, pages 127–138. IEEE.

[Prinz, 1999] Prinz, W. (1999). Nessie : An awareness environment for cooperative settings. In Bodker, S., Kyng, M., and Schmidt, K., editors, *ECSCW '99*, pages 391–410.

[Rai, 2013] Rai, R. (2013). *Socket. IO Real-time Web Application Development*. Packt Publishing Ltd.

[Renger et al., 2008] Renger, M., Kolfschoten, G. L., and de Vreede, G.-J. (2008). Challenges in collaborative modeling : A literature review. In *Advances in Enterprise Engineering I*, pages 61–77. Springer.

[Rittgen, 2008] Rittgen, P. (2008). Coma : A tool for collaborative modeling. In *CAiSE Forum*, volume 344, pages 61–64.

[Rivera et al., 1996] Rivera, K., Cooke, N. J., and Bauhs, J. A. (1996). The effects of emotional icons on remote communication. In *Conference companion on human factors in computing systems*, pages 99–100. ACM.

[Saito and Shapiro, 2005] Saito, Y. and Shapiro, M. (2005). Optimistic replication. *ACM Computing Surveys (CSUR)*, 37(1) :42–81.

[Schmidt, 2002] Schmidt, K. (2002). The problem with 'awareness' : Introductory remarks on 'awareness in cscw'. *Computer Supported Cooperative Work (CSCW)*, 11(3-4) :285–298.

[Software, 2015] Software, L. (2015). Lucid chart. https ://www.lucidchart.com.

[Sohlenkamp and Chwelos, 1994] Sohlenkamp, M. and Chwelos, G. (1994). Integrating communication, cooperation, and awareness : the diva virtual office environment. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 331–343. ACM.

[Steinmacher et al., 2013] Steinmacher, I., Chaves, A., and Gerosa, M. (2013). Awareness support in distributed software development : A systematic review and mapping of the literature. *Computer Supported Cooperative Work (CSCW)*, 22(2-3) :113–158.

[Thum et al., 2009] Thum, C., Schwind, M., and Schader, M. (2009). Slim : A lightweight environment for synchronous collaborative modeling. In *Model Driven Engineering Languages and Systems*, volume 5795 of *Lecture Notes in Computer Science*, pages 137–151.

[Tran et al., 2005] Tran, M. H., Yang, Y., and Raikundalia, G. K. (2005). Supporting awareness in instant messaging : an empirical study and mechanism design. In *Proceedings of the 17th Australia conference on Computer-Human Interaction : Citizens Online : Considerations for Today and the Future*, pages 1–10. Computer-Human Interaction Special Interest Group (CHISIG) of Australia.

[Vertegaal, 1997] Vertegaal, R. (1997). Towards conveying joint attention states. In *Workshop on Awareness in Collaborative Systems in Conference on Human Factors in Computing Systems (CHI'97)*.

[Vertegaal et al., 1997] Vertegaal, R., Velichkovsky, B., and Van Der Veer, G. (1997). Catching the eye. *SIGCHI Bulletin*, 29(4) :87–92.

[Viégas and Donath, 1999] Viégas, F. B. and Donath, J. S. (1999). Chat circles. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 9–16. ACM.

[Ye et al., 2009] Ye, E., Neiman, L., Dinh, H. Q., Liu, C., et al. (2009). Secondwatch : A workspace awareness tool based on a 3-d virtual world. In *Software Engineering-Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*, pages 291–294. IEEE.

[young Bang et al., 2012] young Bang, J., Popescu, D., and Medvidovic, N. (2012). Enabling workspace awareness for collaborative software modeling. In *The Future of Collaborative Software Development at the ACM Conference on Computer Supported Cooperative Work (FutureCSD)*.

[Zhang et al., 2015] Zhang, Y., Chong, M. K., Müller, J., Bulling, A., and Gellersen, H. (2015). Eye tracking for public displays in the wild. *Personal and Ubiquitous Computing*, 19(5-6) :967–981.

[Zhu, 2003] Zhu, H. (2003). Some issues of role-based collaboration. In *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on*, volume 2, pages 687–690. IEEE.