MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE
LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et de l'Informatique
Département de Mathématiques et d'Informatique
Filière : Informatique

MEMOIRE DE FIN D'ETUDES
Pour l'Obtention du Diplôme de Master en Informatique
Option : **Ingénierie des Systèmes d'Information**

THEME :

Un Modèle De Coordination Pour L'édition Collaborative Mobile

Etudiant(e) : « **Hamou Maamar Sabah** »

Encadrant(e) : « Mechaoui Moulay Driss »

Résumé

Le développement rapide des fonctionnalités des dispositifs mobiles, du point de vue matériel et logiciel, offre de nouvelles opportunités pour les environnements paire-à-paire et mobile. Ces fonctionnalités pouvant être utilisée pour fournir des nouveaux services pour les plateformes sociales/collaboratives utilisant des réseaux à courte portée. Parmi ces services, nous nous intéressons à l'édition collaborative des objets partagés par des utilisateurs nomades afin de les manipuler à volonté. Cependant, le maintien de la cohérence des données dans un environnement mobile ou paire-à-paire reste un problème difficile en raison de l'ordre arbitraire de la réception des mises à jour, et de la gestion de la durée de vie de la batterie.

Notre objective est de proposer un modèle collaboratif mobile à faible coût et flexible basé sur l'approche de Transformée Opérationnelle (OT) pour synchroniser les travaux d'édition collaboratives de manière totalement décentralisée. Ce modèle prend en charge les groupes dynamiques où les utilisateurs mobiles peuvent joindre ou quitter le groupe de collaboration à tout moment.

Mots-clés : Collaboration mobile, Transformée Opérationnelle (OT), convergence.

Dédicaces

Je dédie ce travail à :

Mes chers parents, qui ont œuvrés durement pour ma réussite et qui ont cru en moi et m'ont soutenu quotidiennement durant l'élaboration de ce projet ainsi qu'à mon encadreur Mr *Mechaoui* qui a su me conseiller et m'encourager jusqu'à ce que je puisse le mener à terme.

Sans oublier ma famille, mon mari, mes amis, mes proches ainsi que toutes les personnes qui ont contribué à la mise en œuvre de ce travail par leurs idées, leurs conseils ou par leur simple présence.

Remerciements

*Avant tout, je remercie **DIEU**, le tout puissant, pour la force, la volonté, la santé et la patience qu'il m'a donnée pour accomplir ce travail.*

*Je tiens à exprimer mon grand respect et ma gratitude à mon promoteur le professeur **Mechaoui**, pour m'avoir honoré en acceptant de diriger mon mémoire, pour son encadrement de qualité, ses précieuses suggestions scientifiques,*

sa présence encourageante,
et sa patience tout au long de ce travail.

mes reconnaissances vont aussi à tous mes enseignants pour leurs apports inestimables en informations indispensables pour ma formation.

Enfin, je remercie chaleureusement toutes personnes ayant contribués de près ou de loin à l'élaboration de ce travail.

J'adresse mes remerciements infinis à mes parents pour leurs soutiens et leurs encouragements.
Une pensée particulière est adressée à tous nos collègues et amis,
Pour terminer, Merci pour tous ceux qui, par leurs remarques et leurs conseils, ont contribué à la réalisation de ce travail.

Liste des figures

Liste des tableaux

Liste des abréviations

Abréviation	Expression Complète	Page
P2P	Peer to Peer pair à pair	1
CSCW	computer supported cooperative work	4
LAN	Réseau à accès local	11
WAN	Réseau à accès Large (e.g internet)	11
CRDT	Commutative Replicated Data Type	18
OT	Transformée Opérationnelle	21
UML	Unified Modeling Language	31
IT	Inclusive Transformation	37
ET	Exclusive Transformation	38
JDK	Java Development Kit	54
SDK	Software Development Kit	54
SQL	Structured Query Language	55
JSON	JavaScript Object Notation – Notation Objet issue de JavaScript	55
OSMDroid	Open Street Map for Android	56

Table des matières

Introduction

Au cours des dernières années, le monde a vécu une évolution majeure des technologies logicielles et matérielles utilisées dans les systèmes informatiques. Ces systèmes ont été conçus pour but de gestion, de communication ... et sont basés généralement sur des architectures centralisées là où on dispose d'un serveur central responsable des calculs, communication et stockage.

Ce type d'architecture atteint leurs limites et peuvent tomber en panne et d'être inaccessible lors des cas creuse, tels qu'une catastrophe, ou dans les lieux publics à forte fréquentation (stade, concert, meeting politiques, …) qui rassemblent beaucoup de gens en même temps et même lieu essayant tous de se connecter à Internet. Cela a donné naissance à des systèmes P2P là où les sites ne sont plus client ou serveur, mais plutôt jouent les deux rôles simultanément. Ces systèmes P2P permettent de désigner les machines et leur interconnexion à un moment donné, avec un ensemble d'utilisateurs jouant chacun le rôle d'un routeur en passant les messages de recherche voire les données vers leurs destinataires.

L'évolution et l'utilisation continus de ces systèmes a fait apparaitre une nouvelle forme de communication en temps réel qui ne nécessite aucune connexion au serveur dite Peer 2 Peer, les systèmes d'édition collaborative essayent de s'adapter aux nouvelles propriétés de cet environnement. La collaboration en temps réel permet à plusieurs utilisateurs de modifier simultanément le même espace de données tout en travaillant sur différents périphériques et / ou à différents emplacements. Tout changement dans cet espace partagé est immédiatement visible pour tous les collaborateurs.

Les éditeurs collaboratifs en temps réel essaient d'imiter la dynamique de travailler ensemble au même endroit et au même moment. Des changements concurrentiels peuvent être

résolus en temps réel. En revanche, dans les systèmes de contrôle de version classiques, cela se produit de manière asynchrone. Le contrôle de version est néanmoins un outil important de collaboration. Il fournit des informations sur la provenance des données et l'historique des données. En outre, il permet d'inspecter les modifications et de revenir aux anciennes versions, par exemple pour supprimer les erreurs ou les modifications indésirables.

Dans ce travail, nous représentons un système d'édition collaborative permettant à un ensemble d'utilisateurs de travailler sur une même Map (cartes géographiques) à partir de plusieurs sites interconnectés par un réseau P2P. Nous représentons aussi le problème de convergence lors la synchronisation intra-objet qui fait référence à la relation temporelle entre différentes unités de présentation d'un lieu dans la carte dépendant du temps ou lors la synchronisation inter-objets qui fait référence à la relation temporelle entre plusieurs lieux, qui peuvent être indépendants du temps ou dépendant du temps.

Le développement rapide des fonctionnalités des périphériques mobiles (matériel, logiciel et connectivité des données) offre de nouvelles possibilités pour les environnements P2P mobiles qui peut être utilisées pour fournir des services basés sur la localisation, et l'édition collaborative à l'aide de réseaux à courte portée. Parmi ces applications, nous nous intéressons principalement aux éditeurs collaboratifs qui fournissent un support informatique pour modifier simultanément des cartes partagées (Map) sur des appareils mobiles. Cependant, le maintien de la cohérence des données dans un environnement mobile reste un problème difficile en raison de l'ordre arbitraire de réception des mises à jour.

Dans ce mémoire, nous proposons une application collaborative mobile à faible coût pour les appareils mobiles qui permet de mettre à jour simultanément une carte partagée sans avoir besoin de coordination centrale. Elle utilise l'approche de la transformation opérationnelle (OT) pour assurer la cohérence de la carte partagée.

Notre application peut être déployée facilement sur les réseaux P2P car elle prend en charge les groupes dynamiques où les utilisateurs peuvent quitter et se joindre à tout moment.

Ce travail représente un pas en avant vers un environnement d'édition collaborative pratique et flexible pour les cartes partagées.

Chapitre 1 Etat de l'art

1.1 Introduction

Dans ce chapitre nous introduisons les systèmes d'édition collaborative en général. Ensuite, nous présentons les systèmes P2P qui permettent le passage à l'échelle contrairement aux systèmes centralisés (client-serveur). Nous parlons aussi du model de cohérence, et des problèmes de divergence, violation de causalité et d'intention.

1.2 Systèmes d'édition collaborative

1.2.1 Historique

Depuis la fin du 20ème siècle, il y a eu une croissance continue des industries de l'informatique et de la communication. La popularité croissante d'internet a accéléré l'intégration des ordinateurs et des réseaux de communication. Au lieu de créer des applications permettant à une personne de poursuivre uniquement un travail individuel, de nombreux chercheurs et développeurs ont commencé à réfléchir à des systèmes permettant aux gens de travailler ensemble. Un nouveau paradigme d'interaction homme-homme via ordinateur a été créé et la discipline du « travail coopératif assisté par ordinateur » (computer supported cooperative work - CSCW) est apparue. Le CSCW est une discipline scientifique qui guide la conception réfléchie et appropriée et le développement des systèmes informatiques pour soutenir le travail en groupe [3]. Pour ce but, les systèmes d'édition collaborative ont été conçus.

1.2.2 Système d'édition collaborative

Un type particulier du paradigme d'interaction homme-homme via ordinateur est le système d'édition collaboratif. Ce type de système permet aux membres physiquement dispersés d'un groupe de composer, visualiser et éditer ensemble un document (ou un 'objet' plus général). Les types de documents pouvant être édités en collaboration incluent le texte, les graphiques bitmap, les graphiques d'objets, les tableurs, le son et la vidéo, etc. Les systèmes d'édition collaboratif peuvent être utilisés pour :

- Collaborer pour : écriture / création, apprendre concevoir du matériel (ordinateurs, bâtiments, voitures, etc.) Ou des logiciels, programmer, réviser et déboguer.
- Enregistrement d'idées pendant les séances de brainstorming.
- Tenir une réunion.

1.2.3 Systèmes d'édition collaborative graphique

Les documents graphiques peuvent être grossièrement classés en deux types : basé sur un objet, ou basé sur une image bitmap.

Un document graphique bitmap (également appelé « raster ») est créé à partir de lignes des pixels de couleurs différentes qui forment ensemble une image. Dans leur forme la plus simple, les bitmaps ont seulement deux couleurs, avec chaque pixel étant soit noir ou blanc. Avec l'augmentation de la complexité, une image peut inclure plus de couleurs, le cas des images de qualité photographique qui peuvent avoir des millions de variétés de couleurs.

Les documents graphiques objet (également appelés « vecteur ») sont construits à l'aide de formules mathématique décrivant des formes, des couleurs et le placement. Plutôt qu'une grille de pixels, un document graphique d'objet se compose d'objets tels que ligne, rectangle, cercle, etc., qui font ensemble une image. Tandis qu'une image bitmap contient des informations sur la couleur de chaque pixel, un document graphique d'objet contient des instructions sures pour savoir où placer chacun des objets. Il est même possible d'incorporer un graphique bitmap dans un document graphique d'objet, ce qui fonctionne de la même manière

que les graphiques hybrides d'objets bitmap. Cependant, ce n'est pas possible d'incorporer des informations sur les objets dans un bitmap.

Bitmap et les graphiques d'objets ont tous les deux des points forts et d'autres faibles. Par conséquent, ils sont utilisés pour différents objectifs. Les systèmes d'édition de graphiques bitmap sont généralement utilisés pour l'édition de photos, la création d'images et comme tableau blanc pour gribouiller des idées. Les systèmes d'édition graphique d'objets sont utilisés pour les cartes géographiques (MAP), pour dessiner des diagrammes de conception et pour dessiner des diagrammes simples dans des documents ou pour des présentations. [4]

1.3 Passage à l'échelle

Les éditeurs collaboratifs auxquels nous nous intéressons doivent supporter un grand nombre d'utilisateurs. Autrement dit, les mises à jour doivent être propagées à un grand nombre de répliques. Dans notre contexte, le nombre de sites n'est pas constant et la topologie d'interconnexion des sites est variable. Les participants à l'édition collaborative peuvent se connecter ou se déconnecter à tout moment, la collaboration ne doit pas être contrainte par une topologie rigide. Les algorithmes de diffusion de groupes, tels que les algorithmes de diffusion totalement ordonnée [5], sont connus pour passer difficilement l'échelle dans ces conditions. Seuls les algorithmes de diffusion à grande échelle tels que les algorithmes de propagation épidémique sont adaptés au passage à l'échelle.

Dans cette approche, chaque site participe à la diffusion. A intervalle régulier, chaque site choisit aléatoirement un ensemble de sites voisins vers lesquels il propage ses mises à jour. Il a été montré que ces algorithmes ont une forte probabilité à propager les mises à jour à tous les sites. Usenet [7] a clairement montré sa capacité à passer à l'échelle. Il interconnecte aujourd'hui des milliers serveurs. Chaque serveur réplique tous les articles disponibles sur l'ensemble du réseau. Périodiquement, les nouveaux articles publiés sur un serveur sont propagés aux serveurs voisins. Chaque site stocke et transfère les articles reçus à ses propres voisins. Malheureusement, ne préserve pas les intentions pour les opérations d'édition de texte.

Le système idéal pour l'édition collaborative massive serait donc semblable à Usenet mais avec une nouvelle règle qui, à la fois, assure la convergence et qui préserve les intentions [8].

1.4 Architecture des systèmes d'édition collaborative

Le problème majeur concernant l'architecture des systèmes avec un document partagé (ou des données) est l'endroit où le document est stocké et comment ce document est maintenu. En général, le stockage du document peut être centralisé, répliqué ou hybride avec les deux approches.

1.4.1 Architecture centralisée

Avec cette architecture, il existe une distinction entre les sites d'édition et le site du

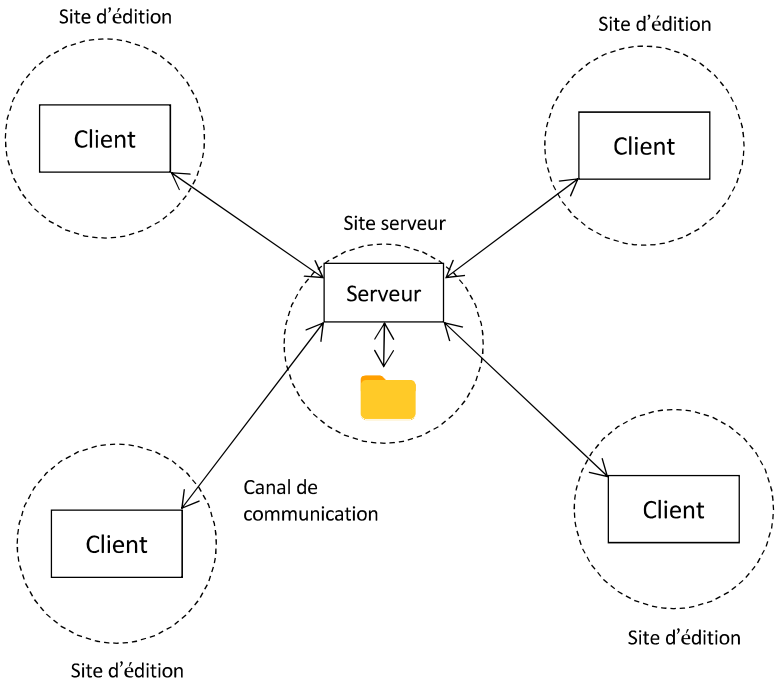


Figure 1 - Une architecture centralisée, avec un processus de serveur unique [8]

serveur. Il peut y avoir plusieurs sites d'édition, chacun exécutant un processus client. Le processus client est chargé de générer des opérations à partir des entrées de l'utilisateur pour mettre à jour le document partagé et afficher le contenu du document partagé sur l'interface utilisateur locale. Il n'y a qu'un seul site serveur. Le document partagé est conservé sur le site du serveur. Le processus serveur est responsable de la mise à jour du document partagé et de

l'envoi du contenu de l'interface utilisateur au client pour l'affichage. Il existe deux stratégies pour mettre à jour le document partagé.

- Il n'y a qu'un seul processus de serveur (comme le montre la figure 1). Puisque seul le processus serveur peut accéder au document, il n'y a pas d'accès simultané au document partagé, donc pas de problème d'incohérence. Cependant, ne pas avoir d'accès simultané est également un inconvénient car cela limite les performances du système puisque, par définition, un seul utilisateur peut accéder au document à la fois. Si un utilisateur émet une opération longue (par exemple, déplacer un objet), aucun autre utilisateur ne peut accéder au document partagé pendant ce temps. Les systèmes utilisent cette architecture pour donner l'illusion d'un accès simultané en ne supportant que des opérations à granularité fine qui peuvent être exécutées dans un temps très court (par exemple, définir la couleur d'un pixel dans un document bitmap). WSCRAWL est un système d'édition graphique collaboratif à base de bitmap avec une telle architecture.
- Il existe plusieurs processus de serveur qui peuvent accéder au document partagé. Actuellement. C'est une approche courante dans les systèmes de bases de données [10]. L'augmentation des performances est obtenue au prix d'éventuelles incohérences causées par l'accès simultané du document partagé. Ces problèmes d'incohérence doivent être résolus en utilisant des méthodes de contrôle de concurrence.

Peu importe qu'il y ait des processus de serveur uniques ou multiples, le principal inconvénient de l'architecture centralisée est que le temps de réponse local peut être long. Cela est dû au fait qu'une fois qu'une opération est générée, son exécution n'est répercutée sur l'interface utilisateur locale qu'après les étapes suivantes :

1. L'opération est envoyée au serveur ;
2. Le serveur exécute l'opération, puis envoie un message (contenant des informations d'interface utilisateur) pour informer tous les clients de la mise à jour ;
3. Le client reçoit le message de réponse du serveur et met à jour l'interface utilisateur locale.

La vitesse d'achèvement de ces trois étapes dépend fortement de la latence du réseau. Dans un réseau où la latence est élevée (par exemple internet), le temps de réponse est long.

1.4.2 Architecture répliquée

Avec l'architecture répliquée [6], il n'y a pas de site serveur. Le processus serveur et le document partagé sont répliqués sur tous les sites d'édition. Par conséquent, chaque site d'édition contient un processus client, un processus serveur et une copie du document partagé. Le processus serveur de chaque site est responsable du maintien de la cohérence de sa copie du document partagé. Les processus du serveur communiquent directement entre eux pour s'assurer que toutes les mises à jour sont disponibles sur tous les sites pour exécution. Cette architecture est illustrée dans la figure 2.

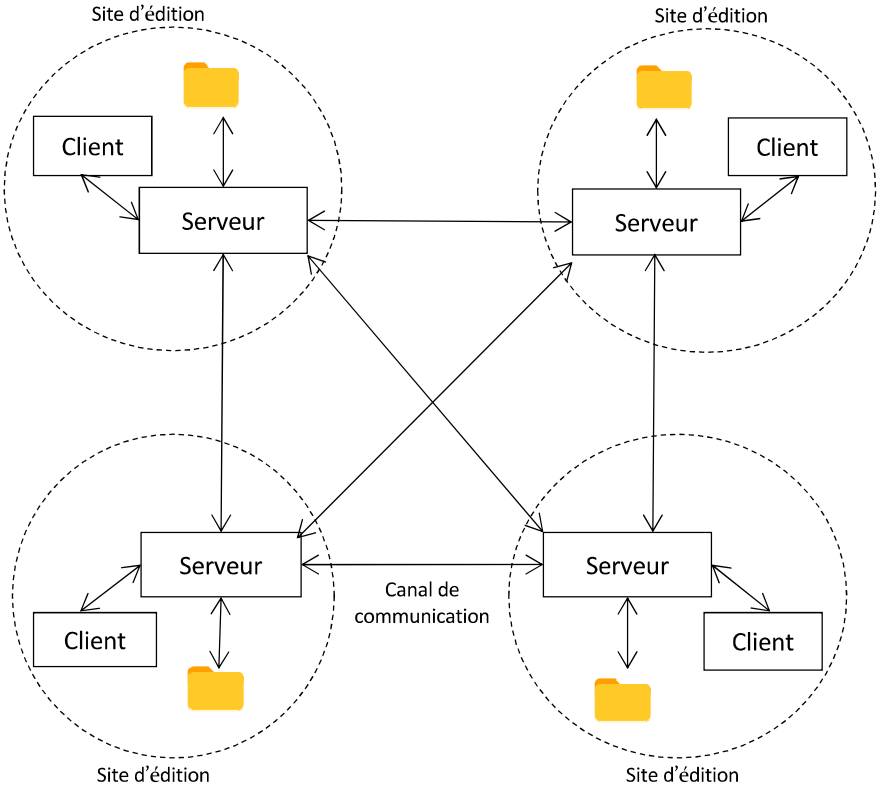


Figure 2 - Une architecture compliquée pour les systèmes de montage collaboratif [8]

Les systèmes d'édition collaboratifs avec une architecture répliquée peuvent fournir un temps de réponse rapide avec une exécution optimiste. Lorsqu'une opération locale est générée, elle est exécutée immédiatement par le processus serveur local et le résultat est instantanément

mis à jour sur son client (sans nécessiter de communication réseau). Le temps de réponse est donc indépendant de la latence du réseau. Cependant, l'inconvénient de cette approche est que peut entraîner des incohérences sur les copies répliquées en raison de la mise à jour simultanée du document partagé effectué par plusieurs processus serveur.

1.4.3 Architecture hybride

En plus de l'architecture centralisée et répliquée, un hybride de ces deux architectures est également possible. Avec l'architecture hybride, chaque site d'édition contient un processus client et un processus serveur plus une copie du document partagé. En outre, il existe également un site serveur contenant le document partagé et un processus serveur.

Lorsque des opérations sont générées, celles qui ne provoquent pas d'incohérence lorsqu'elles s'exécutent simultanément (c'est-à-dire des opérations de création) peuvent être exécutées localement avant d'être envoyées à des sites distants pour exécution (y compris le site serveur). Cela produira un bon temps de réponse.

Cependant, d'autres opérations (toutes les opérations de mise à jour) doivent être exécutées via le site du serveur pour éviter toute incohérence. Le temps de réponse pour ces types d'opérations dépend toujours de la latence du réseau.

1.5 Réplication

Comme pour tous les systèmes d'interaction homme-homme via ordinateur, les systèmes d'édition collaborative graphique peuvent aussi être asynchrones ou synchrones.

1.5.1 Systèmes d'édition collaborative synchrones

Les systèmes d'édition collaborative synchrone ou en temps réel permettent à plusieurs utilisateurs d'éditer le même document en même temps à partir de différents sites. Ces systèmes sont hautement interactifs, où les modifications apportées au document sont automatiquement mises à jour sur tous les sites instantanément. Il existe de nombreuses variétés de systèmes d'édition collaboratifs synchrones, qui se distinguent par leurs caractéristiques.

Dans certains systèmes, le temps de réponse dépend de la latence du réseau sous-jacent, alors que les autres systèmes fournissent un temps de réponse rapide indépendant du réseau. Certains systèmes sont conçus uniquement pour fonctionner dans des réseaux locaux à faible latence (LAN). Ces types de systèmes ne fonctionneraient pas très bien dans les réseaux étendus (WAN) avec une latence plus élevée et non déterministe. Une autre caractéristique est la restriction d'édition placée par le système.

1.5.2 Systèmes d'édition collaborative asynchrones

Avec les systèmes d'édition collaboratifs asynchrones, les documents sont stockés dans un référentiel/base de données central. Avant de pouvoir éditer un document, une copie est effectuée depuis un référentiel central vers le site local. Les utilisateurs éditent leur propre local copie indépendante des autres utilisateurs.

Après quelques modifications, la mise à jour est déclenchée spécifiquement pour mettre à jour le document ou l'enregistrer en tant que version différente dans le référentiel. Des exemples de systèmes d'échantillonnage sont cvs, instant update et prep.

Pour ce type de système, les modifications simultanées dans un seul fichier ou dans toute une collection de fichiers seront logiquement en conflit les unes avec les autres. La solution la plus courante consiste à indiquer l'occurrence d'un conflit et le laisser aux utilisateurs pour le résoudre [2].

1.5.3 L'approche optimiste :

Dans l'approche optimiste, les opérations d'écriture peuvent être effectuées en parallèles sur l'ensemble des répliques. Chaque utilisateur manipule sa copie indépendamment des autres. Les modifications effectuées sur un site ne sont pas envoyées immédiatement vers les autres. Cependant, lorsqu'elles sont diffusées, elles peuvent s'exécutées en parallèle engendrant ainsi des situations de conflit. Dans cette approche les états des copies ne sont pas stables au cours des modifications mais doivent finir par avoir la même valeur une fois les modifications seront effectuées sur toutes les répliques [6].

Ce mécanisme de réplication peut s'appliquer facilement sur les environnements p2p et permet également au système de passer rapidement à l'échelle.

Dans le but d'assurer la cohérence entre les répliques, il est certain que ces systèmes utilisent des mécanismes pour résoudre les conflits et permettre à l'ensemble des répliques de converger vers la même valeur. Dans le cadre d'une édition collaborative, ces mécanismes peuvent suivre différentes approches. Dans ce rapport, on s'intéresse à l'approche de transformée opérationnelle qui propose une conversion préalable des opérations concurrentes avant leur exécution [6].

1.6 Cohérence des données répliquées

Pour répondre à l'exigence de haute réactivité, l'exécution de l'opération optimiste sur une architecture répliquée est utilisée. Ainsi, lorsqu'une opération est générée, elle est exécutée immédiatement sur la réplique du document partagé, puis diffusée sur des sites distants et exécutée dans sa forme d'origine à son arrivée. Il y a trois problèmes d'incohérence qui se manifestent dans les systèmes d'édition collaborative.

1.6.1 Système d'édition collaboratif cohérent

Un système d'édition collaboratif est dit cohérent s'il maintient toujours les propriétés suivantes :

1. Convergence : lorsque le même ensemble d'opérations a été exécuté sur tous les sites, toutes les copies du document partagé sont identiques.
2. Préservation de causalité : pour toute paire d'opérations *oa* et *ob*, si *oa* est lancée avant *ob*, alors *oa* est exécuté avant *ob* sur tous les sites.
3. Conservation de l'intention : pour toute opération o, les effets de l'exécution de *o* à tous les sites sont identiques à l'intention de *o*, et l'effet de l'exécution de o ne change pas les effets des opérations indépendantes.

1.6.2 Convergence

Dans un système de réplication optimiste, les opérations ne sont pas forcément reçues dans le même ordre sur toutes les répliques, même si la causalité est respectée. En effet, deux opérations peuvent être concurrentes, c.-à-d., aucune des deux ne précède causalement l'autre.

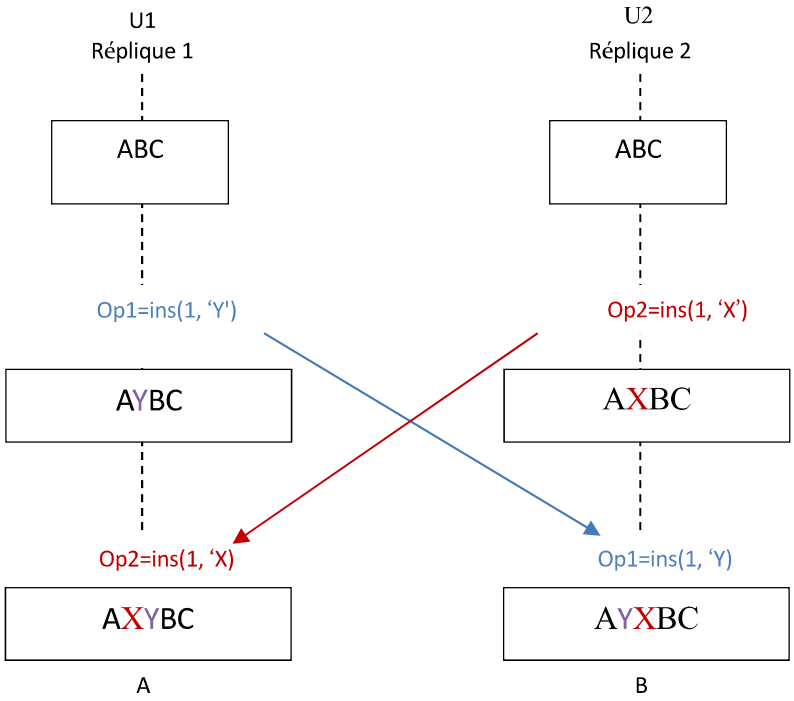


Figure 3 - Exécution concurrente

La figure 3 illustre une situation de concurrence. Les opérations op1 et op2 ont été générées sur les répliques 1 et 2. Intuitivement, deux opérations sont concurrentes si elles ont été générées en même temps. C.-à-d., si chacune a été générée avant la réception de l'autre. Supposons que 2 utilisateurs u1 et u2 modifient un objet en parallèle (**figure 3**), U1 crée une opération op1 alors qu'en parallèle u2 insère une ligne au même endroit avec une opération op2.

Ces deux opérations sont concurrentes, il existe donc deux ordres d'exécution possibles :

- Si le système exécute op1 puis op2 on obtiendrait alors un document contenant : 'AXYBC'
- Si l'ordre d'exécution est op2 puis op1, on obtiendrait le document contenant : 'AYXBC'

Suivant l'ordre de réception, on obtient deux valeurs différentes pour le document. Les deux répliques n'ayant plus le même contenu, alors le système n'assure pas la convergence.

Donc, tout résultat final divergent devrait être interdit dans les systèmes d'édition collaborative où la cohérence des résultats finaux est requise.

1.6.3 Causalité

En raison de latence de communication non déterministe, les opérations peuvent arriver et être exécutées hors de leur ordre de cause à effet naturel.

Comme le montre la **figure 4**, l'opération op2 est générée après l'arrivée de op1 sur le site 2; l'effet d'édition de op1 sur l'objet partagé a été visualisé par l'utilisateur sur le site 2 au moment où op2 est généré. Par conséquent, op2 peut dépendre de op1.

Cependant, puisque op2 arrive et est exécuté avant op1 sur le site 3, une confusion peut survenir sur le système ainsi que sur l'utilisateur sur le site 3. Par exemple, op1 est une opération de création pour créer un objet g et op2 est une opération de déplacement vers déplacez g. Si op2 est exécuté avant op1, comme dans le site 3, alors op2 se référera à un objet qui n'existe pas.

Par conséquent, une exécution hors de l'ordre causal devrait être interdite pour des raisons de correction du système et pour répondre aux exigences d'interaction synchronisée entre plusieurs utilisateurs dans de nombreuses applications.

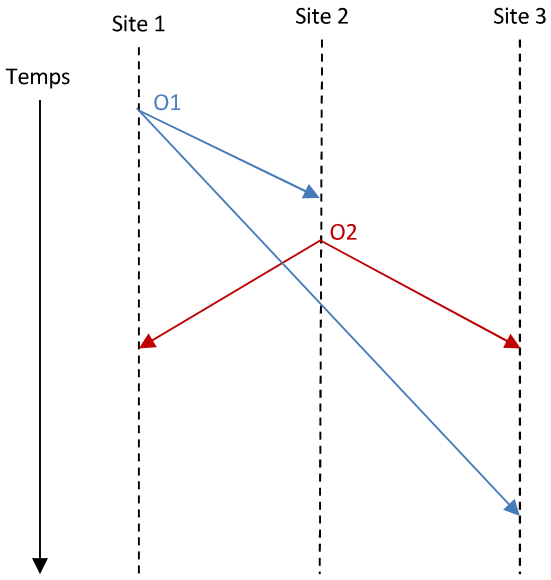


Figure 4 - Arrivé hors l'ordre de cause

1.6.4 Préservation de l'intention

Des opérations conflictuelles peuvent être générées pour remplacer le même attribut du même objet par des valeurs différentes. En conséquence, les effets de certaines opérations ne sont pas conservés.

Par exemple, dans la **figure 4**, si op1 et op2 sont des opérations de déplacement pour déplacer le même objet, g, vers différentes positions, x et y respectivement. Puisque op1 et op2 déplacent le même objet g vers deux positions différentes, il est impossible de prendre en compte leurs effets conflictuels dans le même objet cible. Donc, soit g sera déplacé vers x, soit il sera déplacé vers y, mais pas les deux.

Une conséquence de cette violation d'intention est que, chaque fois qu'il y a un conflit, seul le travail d'un utilisateur peut être préservé. Par conséquent, lorsqu'un conflit survient, les utilisateurs peuvent ne pas voir une image cohérente et explicite de ce que les autres utilisateurs ont fait ou de leurs intentions, et ils ne peuvent donc pas prendre les mesures appropriées pour résoudre leur conflit.

1.7 Conclusion

Dans ce chapitre, nous avons donné une vue globale sur les éditeurs collaboratifs travaillant sur un environnement P2P. Nous avons aussi présenté la notion de réplication et le modèle de cohérence cci qui permet d'évaluer la cohérence des algorithmes développés dans le domaine d'édition collaborative.

Chapitre 2 Edition collaborative sur les objets

2.1 Introduction

Dans ce chapitre, nous décrivons brièvement les algorithmes CRDT et OT que nous allons évaluer et nous précisons pour chaque algorithme leurs avantages et inconvénients pour des applications mobiles. Tous les algorithmes évalués sont conçus pour l'édition collaborative d'un document dont les opérations qui peuvent être exécutées sont l'insertion et la suppression de caractères.

2.2 Approches d'édition collaborative

De nombreuses approches d'édition collaborative en P2Pont été proposées dans la littérature. Peu de ces approches modélisent des objets complexes tels que des arbres ou des graphes. La grande majorité des éditeurs traite le cas des mots, qui sont la forme la plus simple de documents textuels, à savoir une suite de caractères. Beaucoup d'éditeurs proposés dans ce cadre rencontrent des difficultés pour assurer la convergence. Généralement, ils voient un texte comme suites de caractères, lignes ou mots, la position d'un objet dépend des positions des objets précédents.

Lors d'une opération d'insertions ou de suppressions concurrentes, calculer les positions des nouveaux objets ou des objets à supprimer devient vite compliqué. Certaines approches élisent un "leader" pour détecter et résoudre les conflits. D'autres approches imposent le verrouillage de la ressource ou un ordre total sur les opérations.

Ces modèles sont adaptés à un modèle P2Pdans lequel il n'y a aucune centralisation. Une autre approche basée sur la notion de transformée opérationnelle modifie au fur et à mesure les opérations concurrentes reçues en fonction de celles déjà traitées. Cette fonction d'intégration

peut être compliquée à concevoir et la propriété de convergence repose sur cette transformation. Ressel [7] a donné un algorithme qui assure la convergence dans ce cadre à condition que la transformée satisfasse deux propriétés. Ces propriétés sont suffisantes, mais on ne sait pas si elles sont nécessaires dans tous les cas. Cette méthode a pour avantage de pouvoir faire converger les documents sans contraintes (ordre total, verrous) et sans pertes de mises à jour, même si les opérations ne sont pas naturellement commutatives.

Par contre, trouver des transformées qui satisfont les propriétés requisse révèle extrêmement difficile, et actuellement, il n'en n'existe aucune qui satisfait les propriétés exigées par l'algorithme de Ressel pour les mots.

L'approche du "Commutative Replicated Data Type" (CRDT) est une autre façon d'aborder le problème. Au lieu de vouloir transformer des opérations naturellement non commutatives, le but est de concevoir des structures de données et des opérations qui commutent. Ainsi pour le cadre des mots on remplace la numérotation consécutive partant du premier caractère à une numérotation relative ou absolue (non changée pendant l'édition). La structure de données a été légèrement compliquée, mais l'algorithme d'édition devient très efficace.

2.3 CRDT "Commutative Replicated Data Type"

2.3.1 Principe

L'idée principale est de trouver des types de données qui commutent "naturellement" par un algorithme de synchronisation utilisant une façon algébrique pour résoudre les conflits.

Grâce à des règles de détection d'incohérences, l'algorithme effectue les opérations cohérentes et garde les autres jusqu'à pouvoir résoudre les conflits. Il se peut que des opérations restent en suspens pendant un temps non borné. Cette approche correspond aux paradigmes suivants qui pourraient caractériser un synchronisateur de fichier :

1. Propager des opérations non conflictuelles.

2. Si des opérations sont en conflit ne rien faire. Pour appliquer cette approche pour un document, que faut-il faire et quels sont les objets qui permettent de le faire.

– Il est nécessaire d'identifier les éléments à modifier. Comme il n'y a plus d'opération d'intégration pour modifier ces valeurs en fonction des opérations précédentes, les identifiants ne doivent pas être ambigus.

– Il est nécessaire d'identifier des objets et des opérations de base avec une notion de commutation. Dans de nombreux travaux, la structure d'arbre non ordonné ou d'ensemble a été choisie. Ainsi, les opérations d'ajout ou de suppression dans un ensemble, commutent.

– Il est nécessaire de pouvoir représenter l'ordre entre éléments indépendamment des opérations de base si on veut pouvoir passer à des applications sur le format XML. Prenons quelques algorithmes publiés.

2.3.2 Algorithmes Utilisant CRDT

2.3.2.1 WOOT

WOOT est le premier algorithme CRDT qui a été proposé. L'idée de base de WOOT est d'identifier chaque élément par une structure unique. La structure est définie en spécifiant l'identifiant du nouvel élément, le contenu de l'élément et les identifiants qui précédent et suivent cet élément.

La Figure 5 présente l'exécution d'une insertion dans WOOT et illustre le problème de la suppression dans cet algorithme. Dans la Figure 5(a), l'utilisateur 1 insère le caractère 'X' à la position 2 et envoie cette opération à un autre utilisateur. L'opération envoyée contient l'identifiant du caractère 'X' et les identifiants qui précèdent et suivent le caractère 'X' lors de son insertion : 'A' et 'B'. WOOT utilise un ordre global pour ordonner les caractères insérés de manière concomitante à la même position.

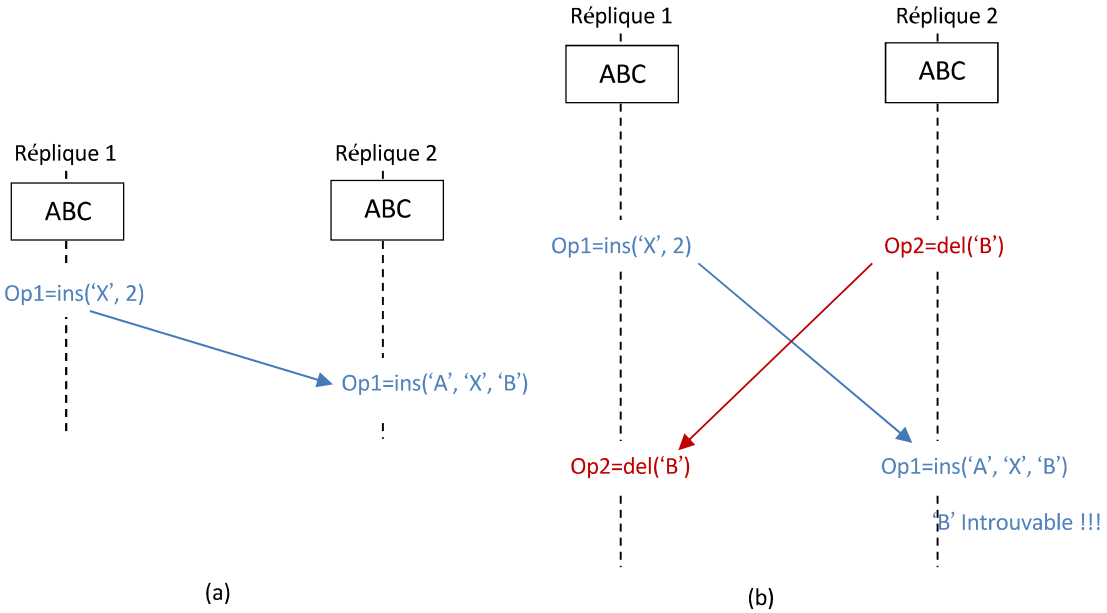


Figure 5- Exécution d'une insertion et d'une suppression dans WOOT

Un des problèmes de WOOT est que l'algorithme doit toujours pouvoir retrouver les éléments suivant et précédent, comme illustré à la Figure 5(b). Pour éviter ce problème, WOOT conserve tous les éléments supprimés (appelés pierres tombales) dans le document. Les éléments ne sont pas supprimés mais seulement rendus invisibles aux utilisateurs. Ceci représente un inconvénient majeur, car le document ne pourra que grossir durant l'édition collaborative, ce qui pourrait devenir un problème face à la capacité mémoire limitée d'un périphérique mobile.

D'un autre côté, WOOT est adapté pour les groupes d'utilisateurs ouverts où les utilisateurs peuvent rejoindre et quitter le réseau. En outre, l'algorithme ne nécessite pas de mécanisme de respect de la causalité des opérations et les opérations sont intégrées dans un ordre quelconque sur chaque copie. Deux solutions optimisées de WOOT ont été proposées : WOOTO et WOOTH. [1]

2.3.3 Limites de l'approche CRDT

L'approche CRDT ne convient pas à tout type d'opérations, en particulier à la suppression de nœud faisant remonter les sous-arbres que nous avions nommée Del2. Nous donnons une justification informelle de ce fait.

L'exemple suivant correspond à deux opérations concurrentes : un site 1supprime une arête *e* dont le père est *ed* et un autre site2 insère une arête *e'* sous celle-ci. Le résultat sur le site 2 donne une arête *ed* qui a comme fils l'arête *e'* et les fils de l'arête *e* qui est supprimée. Sur le site 1, exécuter l'opération d'ajout de l'arête *ed* ne peut se faire car l'identifiant *e* a disparu et l'identifiant de son père n'est pas connu.

2.4 Transformée Opérationnelle (TO)

2.4.1 Définition :

L'approche des **Transformées Opérationnelles** (plus brièvement TO) est un mécanisme proposé par la communauté des éditeurs collaboratifs asynchrones pour répondre aux problèmes de cohérence causés par les opérations concurrentes. Cette méthode s'applique sur un ensemble de nœuds répartis sur un réseau et possédant chacun une copie du document partagé. L'approche TO est considérée comme un mécanisme de réplication optimiste qui permet au système de résister aux mises-à-jour simultanées de l'objet partagé même si ces mises à jour sont diffusées dans un ordre quelconque vers l'ensemble de répliques. Le système est donc mené vers un état stable où toutes les mises-à-jour sont effectuées correctement et par conséquent toutes les répliques sont identiques et convergent vers la même valeur **[6]**.

Comme son nom l'indique, cette approche s'appuie sur la transformation des opérations afin de garantir une cohérence. Ceci est concrètement réalisé grâce à un mécanisme basé principalement sur deux composants qui sont :

Une procédure d'intégration : son rôle est de garantir la diffusion et la réception des opérations de mises-à-jour ainsi que leur exécution. Chaque opération générée ou exécutée localement doit être stockée dans un journal (appelé aussi histoire ou Log). L'algorithme d'intégration doit assurer la gestion des opérations locales et distantes sur lesquelles les transformations vont être effectuées à l'aide d'une fonction de transformation **[6]**.

La fonction de transformation : cette fonction se présente comme un ensemble de procédures assurant la fusion des opérations de mises-à-jour parallèles tout en tenant compte de l'historique des opérations précédentes. En d'autres termes, cette fonction doit prendre en

considération l'impact des opérations déjà d'exécutées sur celle qu'on veut appliquer au niveau d'une réplique afin d'appliquer les bonnes transformations pour l'obtention d'une édition cohérente. Cet ensemble de procédures est spécifique pour chaque type de données **[6]**.

Afin d'assurer la cohérence des répliques, la fonction de transformation doit vérifier les 3 critères principaux du modèle CCI indiqué dans l'état de l'art qui sont : la causalité, la convergence et la préservation de l'intention. Ces critères sont regroupés pour former un modèle de cohérence appelé modèle CCI.

2.4.2 Les conditions de la convergence :

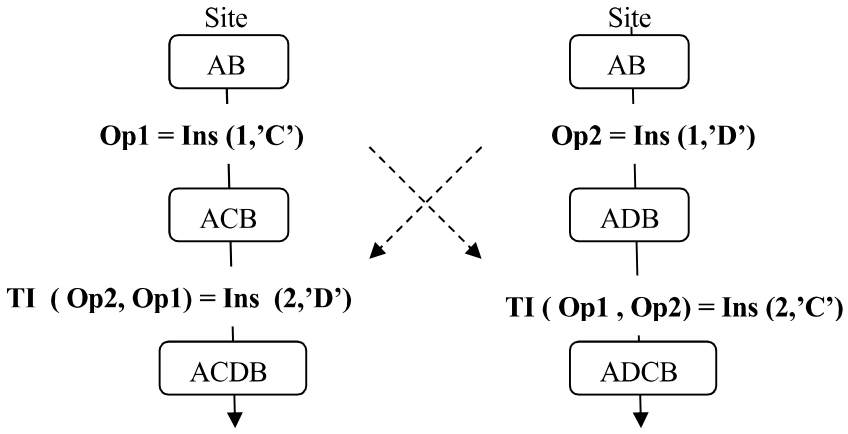


Figure 6- Situation de divergence des répliques.

Il arrive souvent que deux utilisateurs tentent d'insérer en même temps et à la même position, un caractère quelconque. Dans cet exemple, les deux utilisateurs exécutent leurs opérations sur le même état initial au même moment. Ensuite, chacune de ces opérations est envoyée vers le 2ème site pour garantir la mise-à-jour de cette réplique.

On applique automatiquement la fonction de transformation inclusive afin d'éviter tout conflit. Dans ce cas, chaque réplique doit faire face à une séquence d'opération avant que le système se met au repos. Cette séquence est dénotée comme suit : $O_1 \circ O_2 \ldots \circ On$. Le symbole « ○ » représente la concaténation entre 2 opérations successives.

Comme le montre la figure 6, la séquence d'opérations op1○ TI (op2, op1) se termine par l'état "**ACDB**" tandis que l'exécution de la séquence op2 ○ TI(op1, op2) donne l'état "**ADCB**". La situation de divergence se présente clairement même si la préservation de l'intention et l'utilisation de la TI ont été établies convenablement.

Pour assurer la convergence des répliques, la fonction de transformation inclusive (TI) doit satisfaire deux propriétés TP1 et TP2 :

- **TP1** : Cette propriété exige que, pour deux opérations O_1et O_2définies sur un même état initial, l'exécution de la séquence d'opérations O_1○ T(O_2, O_1) produit le même état final que celui après l'exécution de O_2○T(O_1, O_2). Donc, quel que soit l'ordre d'exécution des deux opérations concurrentes, le résultat reste le même.

```
IT (Ins (p1, c1, id1), Ins (p2, c2, id2)):
        si ((p1<p2) ou (p1 = p2 et id1<id2))  alors
                        retourner Ins(p1,c1)
        sinon
                        retourner Ins(p1+1,c1)
        finsi;
```

Afin de garantir que cette opération soit faite de la même manière sur toutes les répliques, une certaine logique devra-t-être poursuite puisque les insertions peuvent se faire mutuellement et que nul ne peut déterminer si l'un des états est meilleur que l'autre.

2.4.2.1 Satisfaction de la propriété TP1

Dans l'exemple précédent, les deux états finaux **"ACDB"** et **"ADCB"** sont divergent, mais il parait difficile de préciser lequel d'entre eux plus correcte que l'autre. Les solutions proposées dans ce contexte visent soit à établir un ordre de priorité sur les opérations ou sur les identifiants des sites, soit à imposer un ordre par rapport aux lettres alphabétiques.

La fonction de transformation TI satisfait parfaitement la condition TP1 sauf dans le cas d'une insertion à la même position, Ressel et al ont montré qu'il est possible d'intégrer cette condition

en ajoutant un ordre sur les identifiants des sites afin de faire converger l'ensemble des répliques [6].

Ainsi, pour deux insertions à la même position, le caractère correspondant à l'identifiant le plus petit sera inséré avant celui de l'autre opération au niveau de la chaine de caractère. La fonction de transformation prend un troisième paramètre indiquant cette priorité d'insertion.

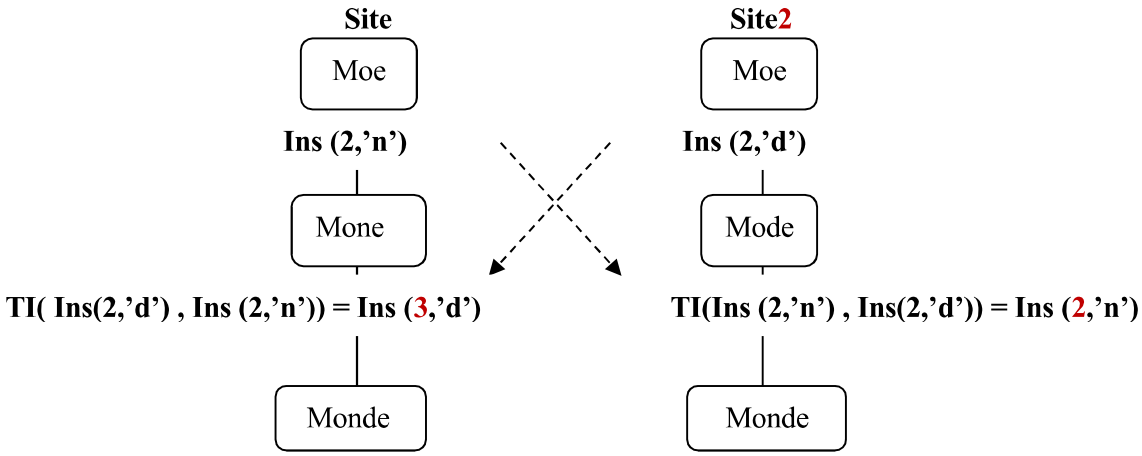


Figure 7 - Convergence de copies avec la satisfaction de la propriété TP1

Dans l'exemple ci-dessus (figure 7), deux opérations d'insertion sont effectuées au même moment et sur le même état initial. Le site 1 génère l'opération **Ins (2,'n')** pour obtenir le mot « **Mone** », en parallèle le site 2 génère l'opération **Ins (2,'d')** pour obtenir « **Mode** ». Les améliorations apportées par Ressel sur l'algorithme de transformation permettent aux deux répliques de converger vers le même état final « **Monde** ». Puisque l'insertion du caractère **d** est générée par le site ayant l'indice 2, la position d'insertion de ce caractère sera incrémentée par 1 au niveau du site ayant l'indice 1.

Il a été démontré, dans l'ensemble des travaux publiés, que la propriété TP1 est nécessaire pour assurer la convergence. Seulement, quand plus de deux opérations concurrentes sont présentes, cette propriété n'est pas suffisante pour garantir une cohérence. La figure 8 suivante décrit un scénario possible pour créer une situation de divergence.

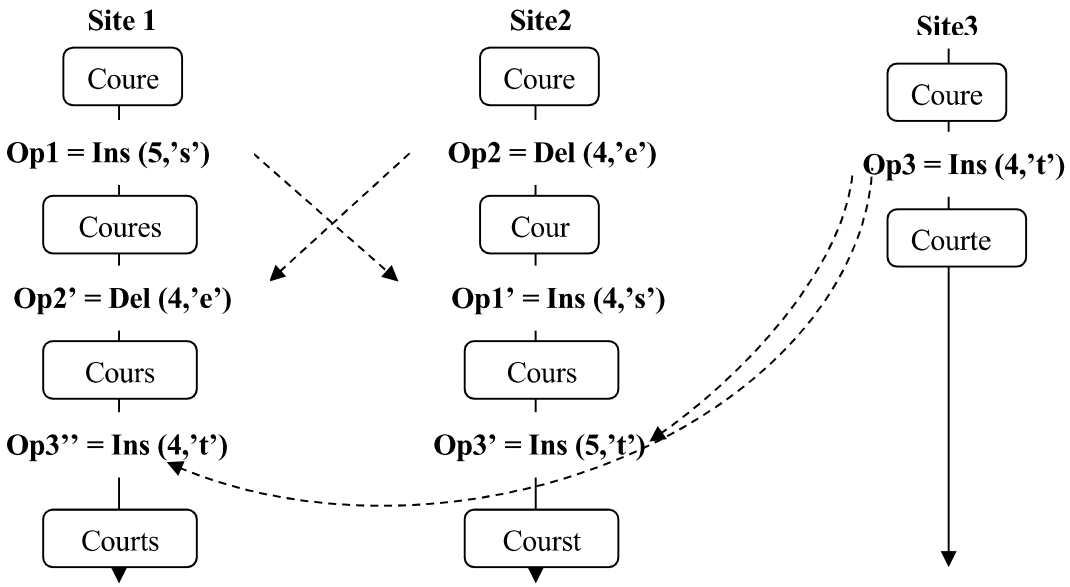


Figure 8 - divergence avec la satisfaction de TP1

Dans cet exemple, les trois utilisateurs essayent d'exécuter concurremment les trois opérations Op1, Op2 et Op3. Lorsqu'on applique l'algorithme de transformation Ins-Del et Del-Ins sur les deux opérations Op1 et Op2 on obtient le même état « Cours ». L'opération Op3 permet d'insérer en parallèle le caractère **t** à la position 4, cette opération est ensuite envoyée vers les autres répliques. Sur le site 1, l'opération Op3 doit être transformée par rapport à Op2' : **Op3'' = Ins (4,'t')** ce qui donne l'état « Courts » en appliquant l'algorithme Ins-Del. Cependant, sur le site 2 l'opération Op3 doit prendre en considération l'effet de Op1' qui est une opération d'insertion, dans ce cas la position d'insertion sera incrémentées selon l'algorithme Ins-Ins : **Op3' = Ins (5,'t').** Puisqu'on applique deux opérations différentes sur le même état on ne peut pas s'attendre à un résultat identique.

2.4.2.2 Satisfaction de la propriété TP2

Une deuxième propriété TP2 doit être respectée pour couvrir l'insuffisance de TP1.

- **TP2** : Pour chaque trois opérations simultanées O_1, O_2et O_3 définis sur le même état, la fonction de transformation T garantie que quel que soit la séquence d'opérations à exécuter, le résultat de cette transformation ne doit pas dépondre de l'ordre selon lequel

ces opérations ont été transformées [9]. Ceci revient à prouver une égalité entre des séquences d'opérations obtenues après la transformation d'un ensemble d'opérations dans différents ordres. Cette propriété est formulée comme suit : T (O_3, O_1◦ T (O_2, O_1)) = T (O_3, O_2◦ T (O_1, O_2))

La transformation de O_3par rapport à la séquence composée de O_2suivie de T (O_1, O_2) doit donner le même état que celui de la transformation de O_3par rapport à la séquence O_1suivie par T (O_2, O_1).

Cette propriété présente jusqu'à présent un grand défi pour les développeurs, toutes les fonctions de transformation déjà publiés ne la font pas satisfaire car elle est très difficile à assurer [15]. Ressel et al ont démontré que ces deux propriétés TP1 et TP2 sont suffisantes pour que la fonction de transformation assure la convergence des copies indépendamment de l'ordre dans lequel les opérations simultanées sont transformées et exécutées [6]. Cependant, pour une édition collaborative, le nombre de paires étant variable, les algorithmes d'intégration développés dans l'approche TO doivent permettre de passer facilement à l'échelle. Jusqu'à l'heure actuelle, plusieurs algorithmes d'intégration ont été proposés mais aucun d'eux ne satisfait parfaitement cette propriété.

2.4.3 Les algorithmes d'intégration :

Pour que la fonction de transformation soit utilisable, plusieurs algorithmes d'intégration ont été développés. Le choix de ces algorithmes dépond dans la plupart du temps du type du réseau sur lequel cet algorithme peut être exploitable ainsi que la structure de données utilisée [9].

Dans une intégration centralisée, des algorithmes tels que SOCT4 et COT permettent la gestion des opérations concurrentes par l'intermédiaire d'un site central en exigeant que l'exécution de ces opérations soit faite dans le même ordre sur tous les sites [9].

Pour une intégration décentralisée, les procédures fréquemment utilisées sont ADOPTED, SOCT2, GOT. Ces procédures ne nécessitent aucun nœud central et peuvent

également supporter un nombre important d'opérations sans imposer un ordre total sur leur exécution.

Le SOCT2 est un algorithme dédié aux systèmes collaboratifs P2P permettant d'assurer au maximum le modèle CCI. Cet algorithme se limite aux structures linéaires telles que les documents textuels.

L'algorithme ADOPTED, qui a été développé principalement par Matthias Ressel, représente une extension de DOPT-algorithme d'Ellis et Gibbs. L'idée de base est de prendre une opération déjà exécutée dans un état passé et la transformer de façon à ce qu'elle peut être appliquée sur l'état actuel.

Cependant, l'algorithme ADOPTED ne remplit pas toutes les transformations possibles surtout lorsqu'il s'agit de traiter une forte activité d'un grand nombre d'utilisateurs. A vrai dire, plus les requêtes distantes attendent leur traitement longtemps, plus les requêtes concurrentes vont s'accumuler sur le Log et plus ça va prendre du temps pour les transformées. Ces algorithmes sont considérés comme incompatibles avec les caractéristiques des réseaux P2P en raison de leur incapacité de passer à l'échelle.

2.5 Systèmes d'édition collaborative sur les objets

2.5.1 Objets (MAP) vs Documents

Les systèmes d'édition collaboratifs basés sur différents types tel que documents, arbre, Map, graphe … ont différents comportements et fonctionnement. Cependant, ils partagent les mêmes problèmes d'incohérence de divergence, de causalité et de violation des intentions. Pour les deux types de systèmes de documents et des cartes géographiques, la divergence peut être résolue par sérialisation, et la violation de la causalité peut être résolue en imposant l'exécution de l'ordre causal. Cependant, il n'existe aucune méthode commune capable de résoudre les violations d'intention pour les deux systèmes, à savoir la transformation opérationnelle est utilisée pour les systèmes de texte et plusieurs versions d'objet sont utilisées pour les systèmes graphiques objet. Pour comprendre ces différences existantes entre les deux systèmes, une comparaison est faite entre la structure du document et les opérations faites.

2.5.1.1 Structure

La manière dont les effets sont conservés dépend de la structure de données du document. Ainsi, le document texte est représenté par une séquence (liste) de caractères simples contrairement à un objet graphique tel que les Maps qui sont représentées par une liste d'objets (Des zones géographiques). La position dans la liste détermine la superposition de ces objets.

Si nous considérons simplement chaque caractère dans un document texte comme un objet (objet caractère), les structures de document entre un fichier texte et une carte graphique deviennent les mêmes. Les deux structures de document sont simplement une liste d'objets (structure de données linéaire) et les opérations utilisées pour éditer ce document peuvent être classées en opérations d'ajout d'objets, de suppression d'objets et de mettre à jour d'objets.

2.5.1.2 Opérations

2.5.1.2.1 Ajout d'objets

Ajouter un objet dans un document texte est une insertion ; on insère un caractère qui va prendre sa propre position en décalant le reste de texte. Le caractère à ajouter risque de prendre une position correcte à cause de l'exécution d'opérations simultanées.

Par contre, une opération d'ajout dans une carte graphique est une création ; on crée un nouvel objet qui va se positionner au-dessus d'un objet existant qui sera toujours gardé dans la même position.

2.5.1.2.2 Suppression d'objets

Une suppression d'un objet fait référence à une suppression pour un document texte, mais elle fait référence à une destruction dans une carte graphique.

Les opérations de suppression rencontrent des problèmes similaires à ceux d'insertion où les caractères à supprimer peuvent être décalés en raison de l'exécution d'opérations simultanées. Par conséquent, les caractères incorrects sont supprimés. Ce problème de violation d'intention peut également être résolu par une transformation opérationnelle.

Cependant, l'opération de destruction d'un objet spécifie son objet cible par les identificateurs uniques des objets. Les identifiants d'objet ne seront jamais modifiés. Par conséquent, en utilisant des identifiants pour déterminer l'objet cible, l'objet correct sera toujours trouvé. Par conséquent, la transformation opérationnelle n'est pas nécessaire.

2.5.1.2.3 Mise à jour d'objets

Les opérations de mise à jour sur un document texte remplacent les valeurs d'attribut de leurs objets ciblés. Contrairement à une carte graphique qui prend en charge les opérations de mise à jour telles que Déplacer, Remplir et Redimensionner. Comme pour la destruction, les objets cibles dont on veut mettre à jour doivent également être déterminés. Ceci est également réalisé en utilisant les identifiants d'objets pour trouver les objets.

Une fois que les objets cibles sont détectés, les opérations de mise à jour sont appliquées pour modifier leurs valeurs d'attribut.

Toutefois, une violation d'intention peut se produire lorsque deux ou plusieurs opérations en conflit sont appliquées au même objet. Par conséquent, plusieurs versions d'objet sont produites pour résoudre les conflits.

2.5.1.2.4 Manipuler les Objets

À partir de l'analyse de la section ci-dessus, il est évident qu'il existe deux méthodes différentes pour trouver des objets auxquels appliquer des opérations.

La première méthode consiste à trouver des objets en utilisant leur position dans le document. Cette méthode sera appelée ciblage par position. Les exigences de cette méthode sont que toutes les positions ont un ordre linéaire et que chaque objet occupe une position différente. Les opérations sont générées avec les positions de leurs objets cibles. Ces positions, spécifiées par les opérations, sont utilisées pour trouver leurs objets cibles sur les sites distants.

Un problème majeur avec le ciblage par position, les positions des objets peuvent changer. Le changement de position de l'objet cible d'une opération, provoqué par l'exécution

d'opérations simultanées, peut entraîner la sélection du mauvais objet. Par conséquent, la transformation opérationnelle doit être utilisée pour garantir que l'objet correct est sélectionné à chaque fois.

La deuxième méthode pour trouver des objets consiste à utiliser l'identifiant unique de chaque objet. Cette méthode sera appelée ciblage par identifiant. L'exigence pour cette méthode est que chaque objet doit avoir un identifiant unique.

Les opérations sont générées avec identifiant de leur objet cible. Ces identifiants, spécifiés par les opérations, sont utilisés pour trouver leurs objets cibles sur les sites distants. La difficulté du ciblage par identifiant est de pouvoir générer simultanément des identifiants d'objet uniques à partir de différents sites.

Le ciblage par identifiant ne souffre pas du même problème que le ciblage par position, où la position de l'opération cible peut être modifiée par l'exécution d'opérations simultanées. En effet, les identifiants d'objet ne peuvent jamais être modifiés par l'exécution d'une opération.

Ces deux systèmes de ciblage ont leurs avantages et inconvénients relatifs. Le ciblage par position présente l'avantage que l'objet cible peut être trouvé facilement (puisque l'emplacement est spécifié). Cependant, il présente l'inconvénient du temps nécessaire pour transformer les opérations et de la complexité de la mise en œuvre de la transformation opérationnelle.

Le ciblage par identifiant a l'avantage de ne pas nécessiter de transformation opérationnelle. Cependant, il présente l'inconvénient de nécessiter un espace mémoire pour stocker les identifiants. En outre, il faut du temps pour rechercher dans la collection d'objets pour trouver l'objet cible.

D'une manière générale, les deux systèmes de ciblage peuvent être utilisés dans des systèmes collaboratifs d'édition de texte ou de graphiques. Cependant, en raison de la nature différente des documents, le ciblage par position est souvent utilisé pour les systèmes d'édition de texte collaboratifs.

2.6 Problème de divergence

Les opérations peuvent arriver et être exécutées sur différents sites dans un ordre différent, ce qui entraîne des résultats finaux différents. Les résultats divergents peuvent être classés en deux types : intra-objet et inter-objet.

2.6.1 Divergence Inter-Objet

Avec la divergence inter-objets, des résultats différents sont produits sur deux objets superposés ou plus sur des sites différents.

Par exemple, soient Op1 et Op2 deux opérations pour créer deux objets qui se chevauchent. G1 et G2 respectivement. En supposant qu'un objet nouvellement créé sera placé au-dessus de tout objet existant. Au site 1, G2 sera au-dessus de G1 Puisque Op2 est exécuté en dernier.

Au site 2, G1 sera au-dessus de G2 Puisque Op1 est exécuté en dernier lieu, donc la divergence inter-objets s'est produite comme montré dans la **Figure 9**.

Apparemment, tout résultat final divergent devrait être interdit dans les systèmes d'édition collaborative où la cohérence des résultats finaux est requise.

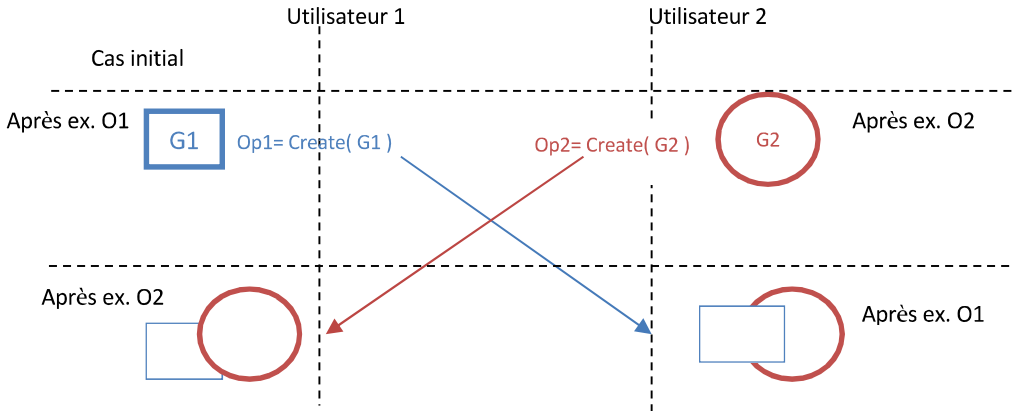


Figure 9 - Problème de divergence Inter-objet

2.6.2 Divergence Intra-Objet

Avec la divergence intra-objet, différents résultats sont produits sur le même objet sur différents sites. Par exemple, laissez les opérations Op1 et Op2 être des opérations de déplacement pour déplacer le même objet, G, vers deux positions différentes, X et Y respectivement. Ces opérations sont appelées opérations en conflit. Des opérations conflictuelles peuvent être exécutées dans différents ordres sur des sites différents (comme le montre **la Figure 10**).

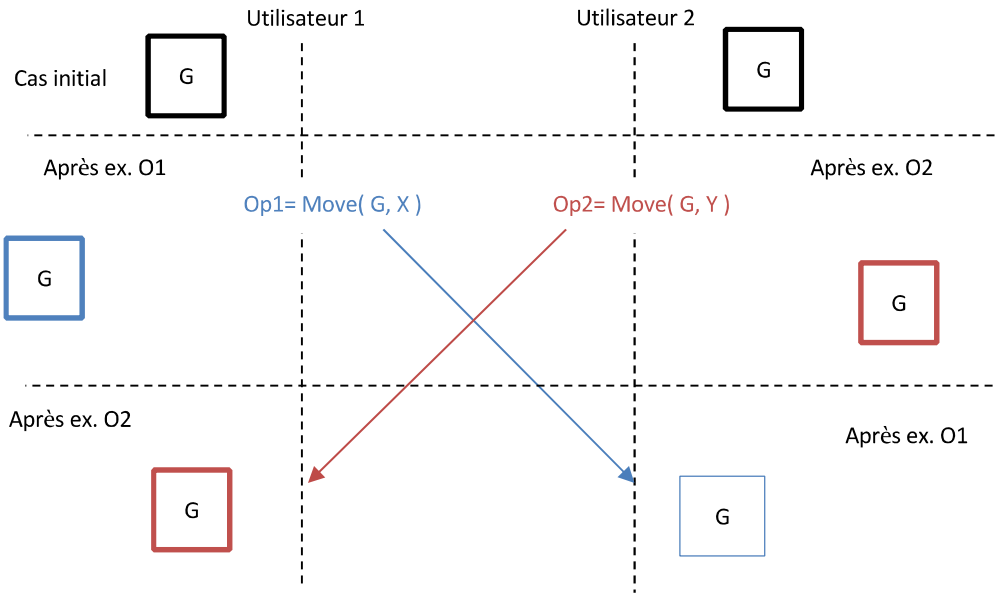


Figure 10- Problème de divergence Intra-Objet et Violation d'intention

2.7 Conclusion

Dans ce chapitre, nous avons donné une vue globale sur les approches utilisées dans les systèmes d'édition collaborative actuels. Nous avons aussi présenté l'approche CRDT et l'approche TO, les algorithmes qui les utilisent et les limites qu'elles posent. Enfin, nous avons détaillé les systèmes d'édition collaborative sur les objets graphiques tels que les Maps et les opérations faites sur ces objets.

Chapitre 3 Conception

3.1 Introduction

Après avoir donné une vue théorique détaillée des différents aspects reliés à l'édition collaborative sur les objets graphiques. Dans une première partie, nous nous proposons d'analyser les différents cas d'utilisation ce qui sert de base pour le passage à l'activité de conception que nous entamerons dans la deuxième partie où nous décriront les vues statiques et dynamique du système en utilisant les diagrammes UML appropriés.

3.2 Protocole de contrôle de la concurrence

Dans ce travail nous proposons une nouvelle application d'édition collaborative mobile pour les cartes (Map) partagé.

Cette application permet à plusieurs utilisateurs équipés d'appareils mobiles et connectés via un réseau spontané (P2P mobile, ad-hoc) de modifier une carte partagée (Map) à tout moment sans utiliser de serveur central pour synchroniser la mise à jour sur la carte partagée. Pour obtenir une disponibilité élevée des données dans les environnements collaboratifs mobiles, une solution directe consiste à utiliser la technique de réplication optimiste. Chaque utilisateur possède sa copie de la carte partagée. L'utilisateur modifie sa copie et envoie ensuite des modifications locales à d'autres utilisateurs (c'est-à-dire d'autres appareils mobiles) pour être intégrer. Cette application repose sur :

- La réplication des cartes géographiques partagées afin de fournir un accès aux données sans contraintes ;
- Le modèle de cohérence basé sur la dépendance causale.

3.3 Modèle d'éditeur des Map collaboratif

3.3.1 Carte géographique (Map) partagée

Pour assurer une édition collaborative sur les MAP, nous considérons un état de carte géographique comme un ensemble de paires de positions *p* et un ensemble de *descriptions*. Une *description* consiste en un identifiant unique et une valeur textuelle. La position *p* est une paire de coordonnées connues sous le nom de latitude et longitude. Une position dans la carte peut contenir plusieurs descriptions afin de prendre en charge des modifications simultanées dans les mêmes positions.

Contrairement aux données partagées classiques en tant que documents, images et listes, une Map n'est pas extensible. Pour traiter cette structure de données linéaire, nous proposons un ensemble amélioré d'opérations pour modifier l'état de Map partagé :

(i) *Add (p, d)* où *p* est la position d'insertion et *d* la description à ajouter à la position *p* ;

(ii) *Del (p, d)* qui supprime la description *d* à la position *p* ;

(iii) *Nop ( )* qui a un effet nul sur la carte géographique partagée.

3.3.2 Requêtes de l'utilisateur

Nous définissons une requête q comme quadruple (*c, r, dep, op*) où

- *c* est l'utilisateur émettant la requête ;
- *r* ∈ N est son numéro de série. Notez que la concaténation de *c* et *r* est définie comme l'identité de q ;
- Le composant *dep* est l'identité de la requête précédente ;
- *op* est l'opération à exécuter sur l'état partagé de la carte.

Si *dep* est nul, la requête ne dépend d'aucune autre requête. Les projections q.c, q.r, q.dep et q.op seront utilisées pour désigner les composantes correspondantes de la requête q. Nous utilisons q, q', q_1, q_2, . . ., pour désigner toutes les demandes. Un tampon d'historique (Log) est

une séquence de requêtes qui est maintenue sur chaque site afin de conserver toutes les requêtes exécutées.

Étant donné un log L,

- L [i] désigne la ième requête de L ;
- | L | est la longueur de L ;
- L [i, j] est le sous-log de L allant de la ième à la jème requête avec 0 <i ≤ j ≤ n - 1 tel que n = | L |.

De plus, nous supposons que les sites sont interconnectés par un réseau fiable.

3.3.3 Relation de dépendance causale

Étant donné un log, une requête peut dépendre des requêtes précédentes en fonction de l'ordre d'exécution. Le suivi de cette dépendance dans un log nous permet d'identifier les requêtes qui doivent être exécutées sur tous les sites selon le même ordre.

Pour résoudre le problème de la causalité-violation (voir Figure 10 du chapitre 2), nous proposons une relation de dépendance minimale indépendante du nombre d'utilisateurs et, par conséquent, permettant ainsi la création des groupes dynamiques.

L'étude de la sémantique d'un objet linéaire telle qu'une Map, modifiée par des opérations d'insertion et de suppression permet de fournir la relation de dépendance suivante dans un log :

3.3.3.1 Définition (Relation de dépendance causale)

Soit L un log où L [i] = $q_i$et L [j] = $q_j$avec j = i + 1.

Nous définissons la relation transitive s → sur L comme suit.

Nous disons que $q_i \xrightarrow{s} q_j$si l'une des conditions suivantes est remplie :

1. q_i.op = Add (p, d), q_j.op = Del (p, d'); (df1)
2. q_i.op = Del (p, d), q_j.op = Add (p, d'); (df2)

Sinon, si q_i s $\overset{s}{\not\rightarrow} q_j$ alors $q_i$et q_j sont indépendants (simultanés).

Nous définissons deux relations de dépendance causale ;

(i) Supprimer une description dépend de la requête qui a ajouté cette description selon (df1) ;

(ii) Ajouter la même description à la même position dépend de la requête qui a ajouté cette description selon (df2).

De plus, il n'y a pas de dépendance entre les demandes de suppression et elles peuvent être exécutées entre elles dans n'importe quel ordre. Dans ces cas, chaque requête doit uniquement stocker l'identité de la requête dont elle dépend directement.

3.3.4 Fonctions de transformation

Notre protocole utilise deux formes de transformation :

(i) La transformation inclusive permet à une opération d'inclure l'effet de toutes les opérations simultanées. Cette transformation est désignée par IT.

(ii) La transformation exclusive nous permet d'exclure éventuellement (selon Définition 1) toutes les demandes qui précèdent la demande en cours. Cette transformation est désignée par ET.

L'algorithme IT est utilisé pour exécuter des requêtes simultanées dans n'importe quel ordre. Tous les cas pour notre IT sont donnés dans l'algorithme1.

Lorsque deux requêtes ajoutent la même description à la même position (elles sont en conflit), un choix doit être fait (lignes 4-5).

Algorithme 1 IT

IT (q_1, q_2) = q_1'
$q_1' \leftarrow q_1$
Choix de q_1**ET**q_2
  Cas :$q_1 = Add_1$**ET** $q_2 = Add_2$
    Si (p2 = p1 **ET** d2 = d1) **Alors** Nop ()
    Sinonq_1'.o ← *Add*(p 1, d1)
  Cas : $q_1 = Add_1$**ET**$q_2 = Del_2$
    SI (p2 = p1 **ET** d2 = d1) **Alors** Nop()
    Sinonq_1'.o ← *Add* (p1, d1)
  Cas : $q_1 = Del_1$**Et**$q_2 = Add_2$
    Si (p2 = p1 **ET** d2 = d1) **Alors**
q_1'.o ← *Del*(p1, d1)
  Cas : $q_1 = Del_1$**ET**$q_2 = Del_2$
    Si (p2 = p1 ET d2 = d1) **Alors** Nop ()
    Sinonq_1'.o ← Del (p1, d1)
fin choix
retourneq_1'

L'IT renvoie la requête inactive dont le paramètre d'opération est Nop, qui a un effet nul sur l'état partagé. D'un autre côté, lorsque deux requêtes sont supprimées à la même position, l'IT retourne la requête inactive Nop. Le reste des cas de l'IT sont assez simples.

L'utilisation de l'algorithme ET nous permet de détecter la relation de dépendance causale entre les demandes selon la Définition 1 à l'intérieur d'un Log. Lorsque ET (q_1, q_2) renvoie "**Indéfini**" alors $q_2 \overset{s}{\rightarrow} q_1$.

Par exemple, lorsque q_1 et q_2 sont successivement une opération d'insertion et de suppression à la même position et la même description, l'élément ajouté par q_1 doit précéder celui ajouté par q_2 (voir Algorithme 2, ligne 7- 8). Tous les différents cas de notre ET sont donnés dans l'algorithme 2.

La figure 11 illustre la solution de proposition pour le problème de divergence (présenté dans le chapitre 2) en utilisant des fonctions de transformation. Dans la 1ère zone, quand O_2 est reçue, elle est transformée inclusivement avec O_1 pour donner :

O_2'=IT(O_1 O_2)=Nop où O_2' n'a aucun effet sur l'état de la carte géographique puisqu'elle est déjà vide.

Algorithme 2 ET

ET (q_1, q_2) = q_1'
$q_1' \leftarrow q_1$
Choix de q_1**ET**q_2
Cas : q_1= Add_1**ET**q_2= Add_2
Si (p1 != p2) **OU** (p1 = p2 **ET** d1 = d2) **Alors**
q_1'.o ← *Add*(p1, d1)
Sinonq_1'.o ← *Add* (p1, d1)
Cas : q_1= Add_1**ET**q_2= Del_2
Si (p1 = p2 **ET** d2 = d1) **Alors**
retourner "Indéfini "
Sinon q_1'.o ← *Add*(p1, d1)
Cas : q_1= Add_1**ET**q_2= Add_2
Si (p1 = p2 **ET** d2 = d1)
returner "Indéfini "
Sinon q_1'.o ← *Del* (p 1, d1)
Cas : q_1= Del_1 **ET**q_2= Del_2
Si (p1 != p2) **OU** (p1 = p2 **ET** d1 = d2) **Alors**
q_1'.o ← *Del* (p1, d1)
Sinonq_1'.o ← *Del* (p1, d1)
fin choix
retourner q_1'

Pour traiter le problème de causalité, quand la zone 3 génère O_3, elle sera transformée avec O_1(ET (O_3, O_1)) et la zone détecte que O_3dépend de O_1. Donc, O_3doit être exécuté après O_1dans toutes les zones.

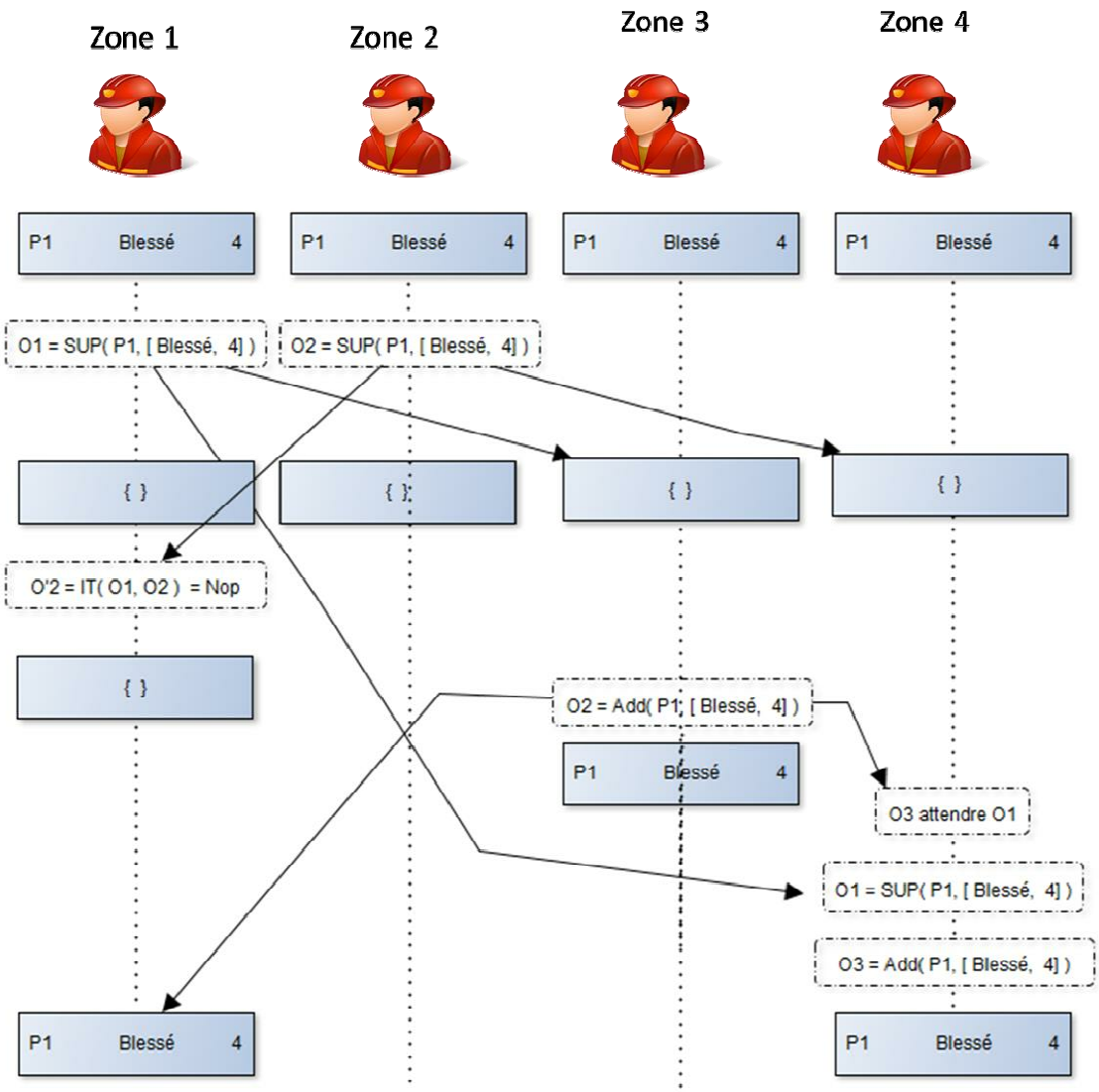


Figure – 11 un scénario de convergence

Quand O_3est reçu dans la zone 4, elle attend l'exécution de O_1pour qu'elle puisse s'exécuter.

Procédure de contrôle

Dans l'approche TO, un éditeur de cartes collaboratif se compose d'un groupe de N sites (où N est variable dans le temps) démarrant une session de collaboration à partir de la même carte initiale. Chaque site stocke toutes les demandes exécutées dans un journal Log.

Génération de demande locale

Lorsqu'une opération op est générée localement, elle est immédiatement exécutée sur son état de carte. La position peut être définie sur la carte partagée de deux manières : à la demande ou automatiquement.

- L'identification à la demande nécessite que l'utilisateur clique sur l'écran du terminal mobile avec son doigt ou son stylet.
- Le point d'accès automatique est disponible pour les appareils intégrant le GPS. Le point d'accès automatique est disponible pour les appareils intégrant le GPS. Il se concentre sur l'emplacement actuel des utilisateurs et déplace automatiquement la carte en fonction du mouvement des utilisateurs.

Une fois la requête $q = (c, r, null, op)$ formée, la fonction de transformation *ET* est appelée afin de calculer la relation de dépendance de q pour maintenir la causalité des cartes partagées. Le résultat de cette transformation q' est envoyé aux autres mobiles pour être intégrer.

Intégration d'une requête distante

Chaque site a l'utilisation de la file d'attente Q pour stocker les requêtes distantes provenant d'autres sites. La requête q générée sur le site i est ajoutée à Q lorsqu'elle arrive sur le site j (avec $i \neq j$).

Deux cas sont à considérer :

(i) Si q : $dep = null$ alors q est concurrent à toutes les requêtes de Log. Dans ce cas, q est causalement prêt.

(ii) Si q : $dep \neq null$ alors il existe une requête Log [k] (avec k $\in \{0,\ldots, n\text{-}1\}$ et n la taille du Log dont q dépend. Si Log [k] n'existe pas, alors q n'est pas causalement prêt.

Quand la requête q sera causalement prête, alors la fonction de transformation *IT* est appelée pour inclure l'effet de cette requête sur le Map partagé.

3.4 La conception de l'application

3.4.1 Méthodologie et approche adoptée

Avant de développer l'application et se lancer à coder, il faut que nous organisions nos idées, les documentions, puis organisions la réalisation en définissant les modules et les étapes de la réalisation. Cette démarche appelé modélisation consiste à créer une représentation virtuelle d'une réalité de telle façon nous faisons ressortir les points dont on s'intéresse. Dans le cadre de ce projet nous avons utilisé la méthodologie UML pour la modélisation des différents diagrammes.

3.4.2 Présentation d'UML

UML, (Unified Modeling Language) est un langage de modélisation unifié permettant de modéliser une application logicielle d'une façon standard dans le cadre de conception orienté objet. Il permet de couvrir le cycle de vie d'un logiciel depuis la spécification des besoins jusqu'au codage en offrant plusieurs moyens de description et de modélisation de l'utilisation système, des acteurs, du, du flot de contrôle internes aux opérations, comportement des objets, des composants d'implémentation et leurs relations, de la structure matérielle et de la distribution des objets et des composants indépendamment des techniques d'implémentation. Il peut aussi être mis à jour selon les besoins.

3.4.3 Avantages d'UML

UML est : Universel ; Adopté par les grandes entreprises ; Muni d'une notation unifiée ; Facile à comprendre ; Adopté par plusieurs processus de développement ; Limite les risques d'erreur ; N'est pas limité au domaine informatique.

La conception détaillée met en œuvre itérativement un microprocessus de construction et c'est en cette phase que l'on génère le plus de volume d'informations. En tant que concepteurs, nous allons élaborer le modèle de conception qui va donner une image « prête à coder » de notre solution. Cette étape se fera par étape afin d'aboutir à un système fonctionnel reflétant une réalité physique.

3.4.4 Diagrammes de cas d'utilisation

Le diagramme de cas d'utilisation a pour but de donner une vision globale sur les scénarios que nous réaliserons dans l'application. C'est un diagramme UML constitué d'un ensemble d'acteurs qui agit sur des cas d'utilisation et qui décrit le comportement d'un système du point de vue utilisateur à travers des actions et des réactions. Il est composé de trois type d'élément :

Acteur : est une entité ayant un comportement telle qu'une personne, un système ou une entreprise qui communique et interagit avec les cas d'utilisation du système.

Système : un élément servant à fixer les fonctions qu'il doit fournir à l'intérieur du système et les limites que le système présente en relation avec les acteurs qui l'utilisent.

Cas d'utilisation : est un ensemble d'actions à réaliser par le système en produisant un résultat observable et intéressant pour un acteur particulier.

3.4.5 Identification des acteurs

Tableau 1 - Les acteurs du système

Acteur	Rôle
Editeur de zone (Utilisateur mobile)	- Authentification - Consultation des contactes - Création de zone - Jointure de zone - Manipulation des Marker (ajout, suppression …) - Chat avec ses contactes

3.4.6 Diagramme de cas d'utilisation globale

Ci-dessous dans la figure 12, nous présentons le diagramme de cas d'utilisation pour la compréhension du fonctionnement du système.

Figure 12- Diagramme de cas d'utilisation globale

3.4.7 Description du cas d'utilisation « consulter contacts »

Tableau 2 - Fiche de description du cas d'utilisation : Consulter contactes

Titre	Cas d'utilisation « consulter contacts » associé à la figure 13
But	Afficher les contacts sur la carte géographique ou dans une liste
Résumé	Le client demande au système la liste des contacts via un menu déroulant, ou les affiche sur le Map avec le positionnement de chacun d'eux
Acteur	L'Utilisateur
Pré Conditions	L'utilisateur est authentifié
Post conditions	Afficher la liste de contacts soit sur Map ou en une liste
Scénario nominal	DEBUT • L'utilisateur se connecte à l'application. • L'utilisateur valide la requête d'affichage des contacts via deux différents boutons correspondants. • Le système affiche selon le bouton cliqué les contacts, sur Map ou dans une liste. • L'utilisateur peut voir ses contacts sur Map ou basculer dans le menu des contacts pour plus de détails. FIN

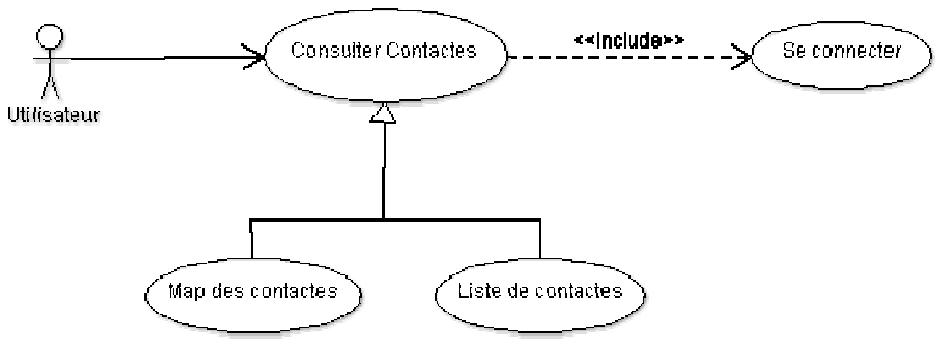


Figure 13 - Diagramme de cas d'utilisation « consulter contacts »

3.4.8 Description du cas d'utilisation créer zone

Tableau 3 - Fiche de description du cas d'utilisation : créer zone

Titre	Cas d'utilisation « Créer Zone » associé à la figure 14
But	Créer une zone
Résumé	L'utilisateur demande la création d'une zone, il est invité à spécifier le schéma et les participants et valider la nouvelle zone.
Acteur	L'Utilisateur
Pré Conditions	L'utilisateur est authentifié et n'appartient à aucune zone
Post conditions	Créer une zone et inviter les participants
Scénario nominal	DEBUT • L'utilisateur se connecte à l'application. • L'utilisateur demande la création d'une zone via un bouton correspondant. • Le système invite l'utilisateur à dessiner le schéma de la zone à créer. • L'utilisateur dessine et valide le schéma de la zone • Le système affiche une liste de contactes et invite l'utilisateur à choisir les participants. • L'utilisateur choisit les participants et valide la création de la zone à créer. • Le système crée la zone et invite les participants à joindre la zone. FIN

Scénarios Alternatifs	• Si l'utilisateur annule le dessin du schéma, le système annule l'opération d'ajout de zone. • En cas où l'utilisateur ne choisit aucun participant et valide la création de zone, le système affiche une erreur et demande à l'utilisateur de choisir au moins un participant avant de valider.

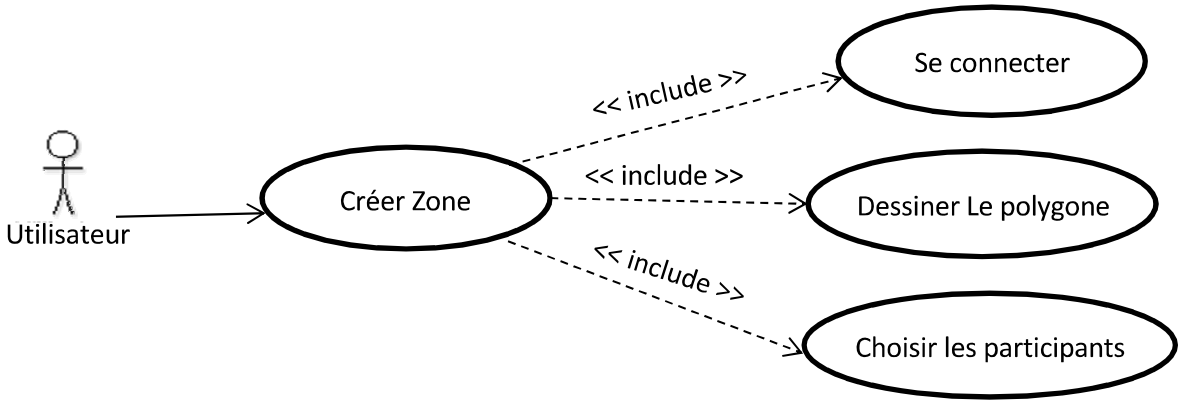
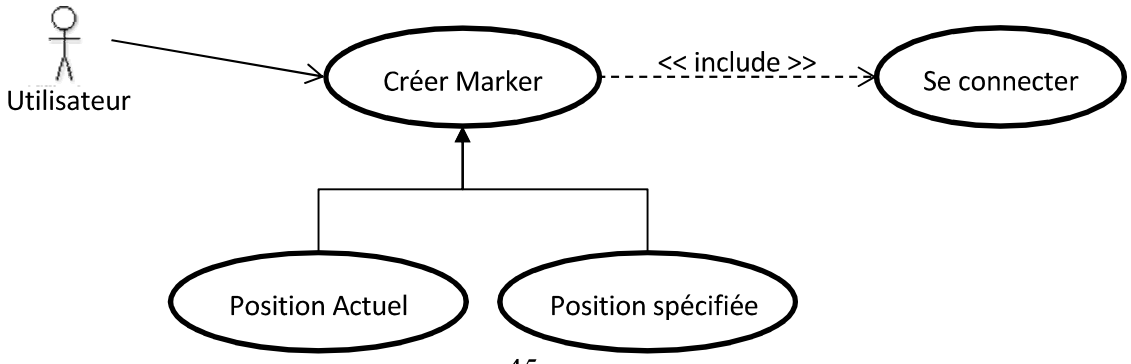


Figure 14 - Diagramme de cas d'utilisation « créer zone »

3.4.9 Description du cas d'utilisation « ajouter Marker »

Tableau 4 - Fiche de description du cas d'utilisation : ajouter Marker

Titre	Cas d'utilisation « ajouter Marker » associé à la figure 15
But	Ajouter un Marker avec sa description et l'afficher sur le Map



Résumé	L'utilisateur clique sur un bouton flottant d'ajout de Marker, le système affiche un dialogue contenant un formulaire à remplir. Une fois la saisie validée, le Marker est créé et diffusé aux autres utilisateurs.
Acteur	L'Utilisateur
Pré Conditions	L'utilisateur est authentifié et appartient à une zone
Post conditions	Créer un Marker et le diffuser aux autres utilisateurs
Scénario nominal	DEBUT • L'utilisateur se connecte à l'application. • L'utilisateur demande l'ajout d'un marker via un bouton correspondant. • Le système affiche un dialogue ayant un formulaire à remplir. • L'utilisateur remplit le formulaire et valide le marker à créer. • Le système crée le Marker et diffuse ses données pour les autres utilisateurs. FIN
Scénarios Alternatifs	• Si l'utilisateur annule l'ajout du Marker, le système fait disparaitre le dialogue. • En cas où l'utilisateur ne remplit pas un champs obligatoire et valide le Marker, le système affiche une erreur et demande à l'utilisateur de saisir avant de valider.

Figure 15 - Diagramme de cas d'utilisation « ajouter Marker »

3.4.10 Diagrammes des séquences

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation UML. Dans ce qui suit, nous présentons le diagramme de séquence selon le diagramme de cas d'utilisation dans notre système.

3.4.10.1 Diagramme de séquence relatif à « se connecter »

A travers ce diagramme, nous allons décrire le scénario du cas d'utilisation « se connecter ». Dans un premier lieu, l'utilisateur remplit le formulaire d'authentification en introduisant son nom et clique sur le bouton « Connexion ». Ensuite le système génère son ID et vérifie s'il existe déjà des données locales pour l'ID généré afin de les charger.

La figure 16 illustre une description détaillée du scénario relatif au cas d'utilisation «se connecter ».

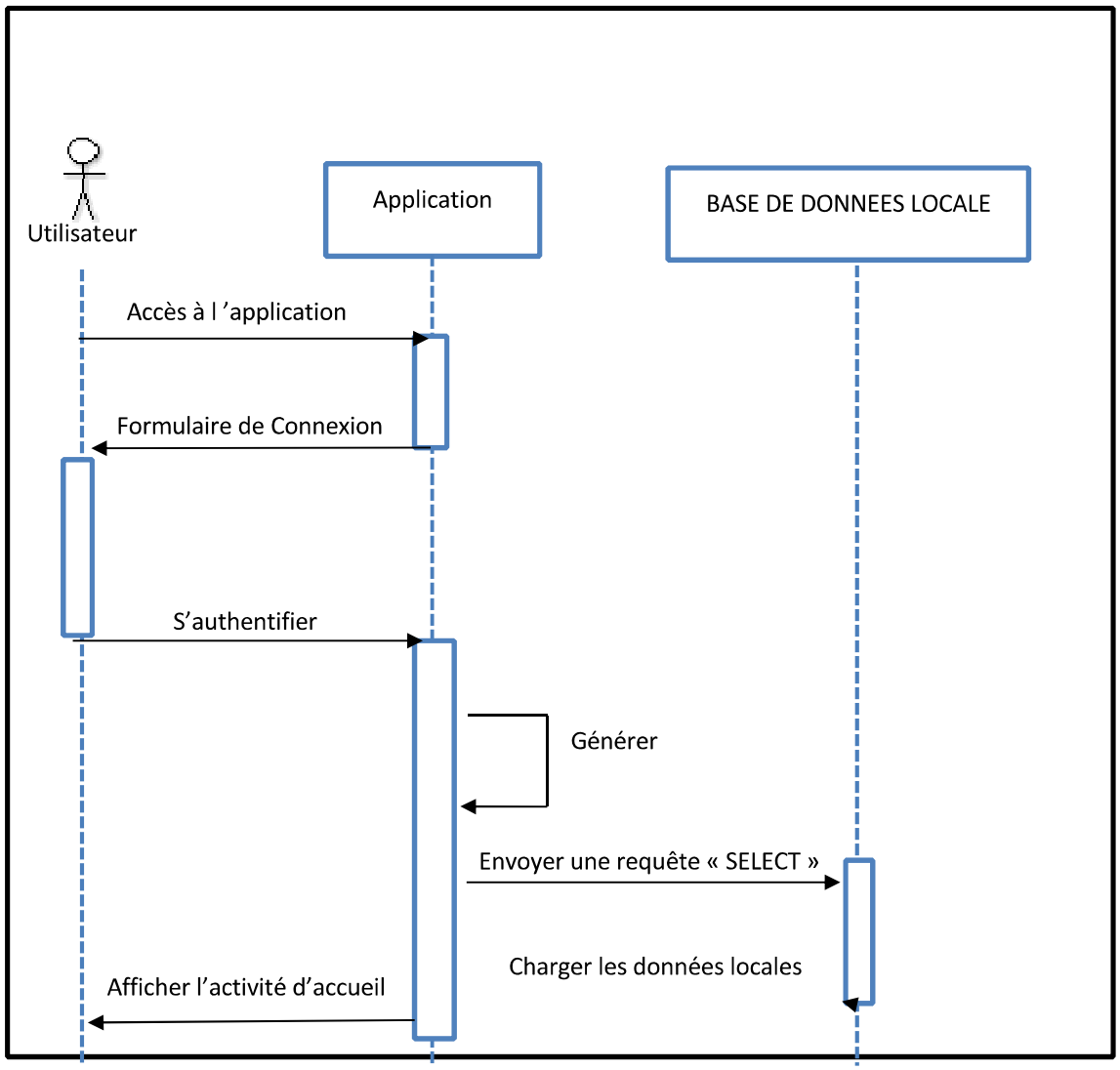


Figure 16 : Diagramme de séquence relatif à « se connecter »

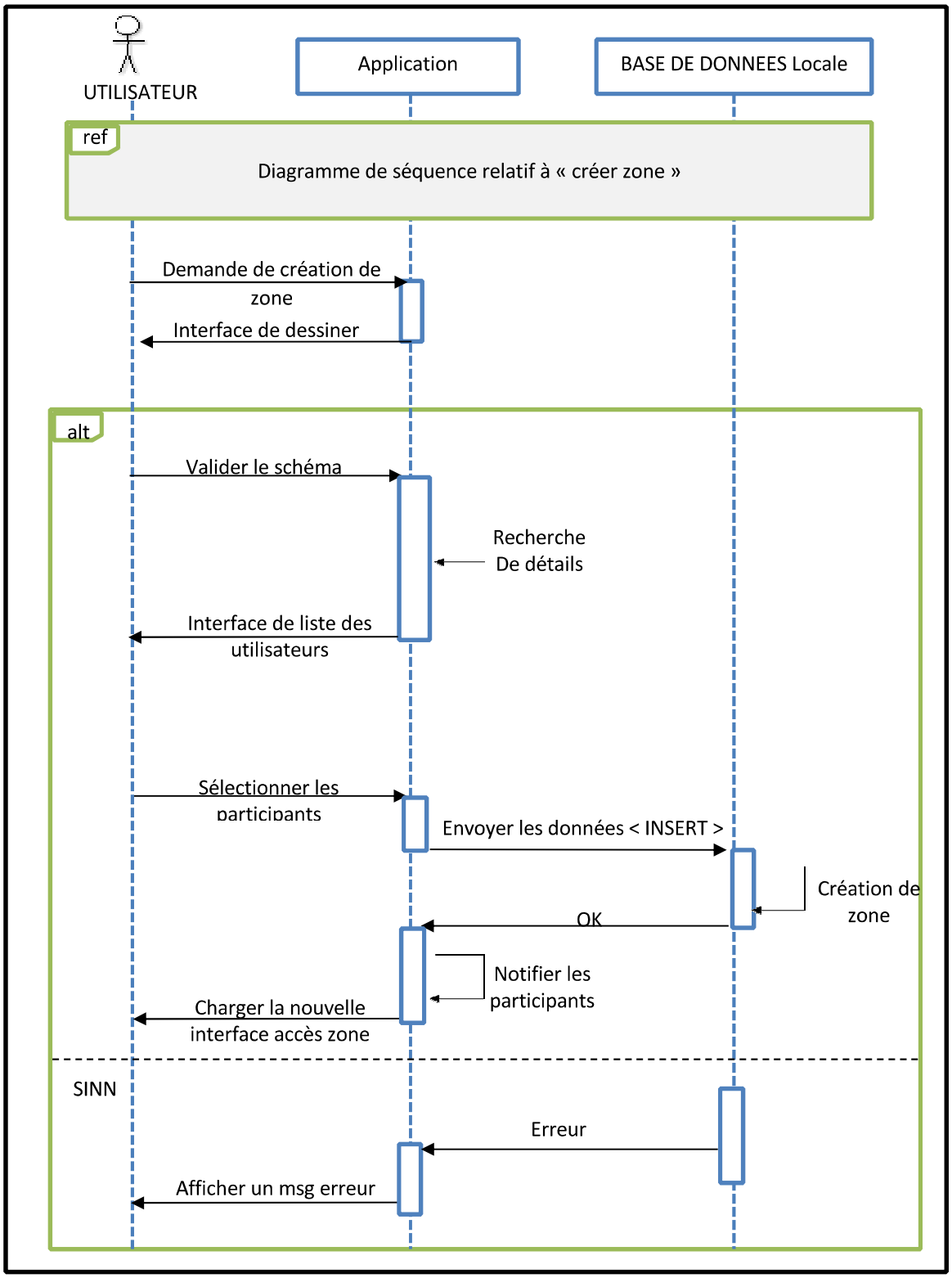


Figure 17 - Diagramme de séquence relatif à « créer zone »

3.4.11 Diagramme de séquence relatif à « créer zone »

A travers ce diagramme, nous allons décrire le scénario du cas d'utilisation « créer zone ». Dans un premier lieu, l'utilisateur demande au système de vouloir créer une zone. Le système lui invite à dessiner le schéma de la zone et choisir les participants à partir de ses contacts. Une fois la création de la zone est validée, le système insert la zone à la base de données locale et notifie les participants de cette zone. La figure 17 illustre une description détaillée du scénario relatif au cas d'utilisation « créer zone ».

3.4.12 Diagramme de séquence relatif à « consulter contactes »

A travers ce diagramme, nous allons décrire le scénario du cas d'utilisation « consulter contactes ». Dans un premier lieu, Le Prospect/Client clique sur l'item « liste devis » à partir du menu principal depuis l'interface graphique, alors l'application lancera une recherche dans la table « User » afin de récupérer la liste des devis trouvés dans la base de données. La figure 18 illustre une description détaillée du scénario relatif au cas d'utilisation « consulter contactes ».

3.4.13 Diagramme de séquence relatif à « localiser les contacts »

A travers ce diagramme, nous allons décrire le scénario du cas d'utilisation « localiser ses contacts ». Dans un premier lieu, L'utilisateur peut visualiser ses contacts à proximité de sa position. Il peut ainsi consulter la liste des utilisateurs sur une carte selon son choix. Ensuite Lorsqu'il sélectionne un marqueur sur la carte les informations correspondantes seront affichées dans une autre interface. La figure 19 illustre une description détaillée du scénario relatif au cas d'utilisation « localiser les contacts ».

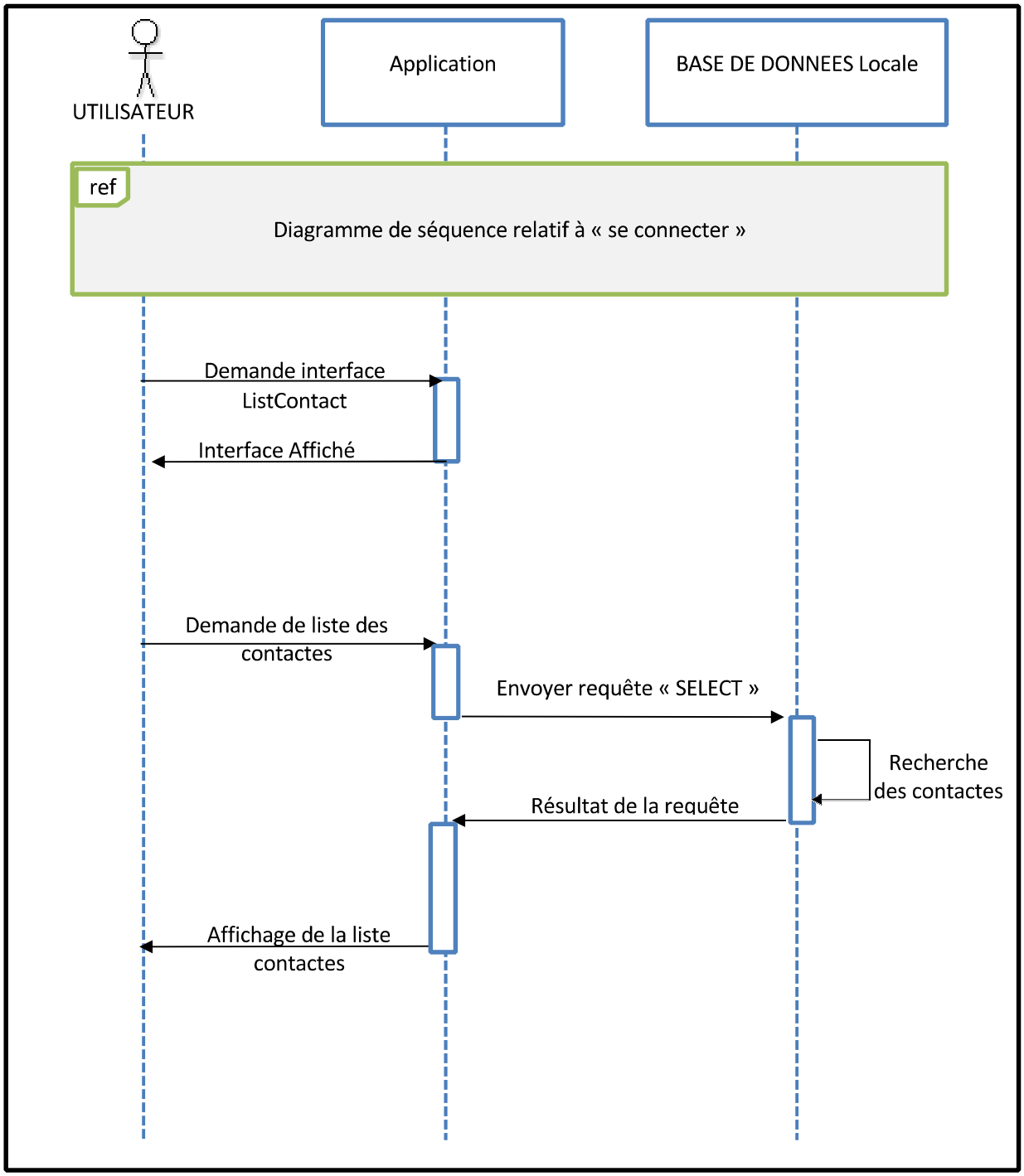


Figure 18 - Diagramme de séquence relatif à « consulter contacts »

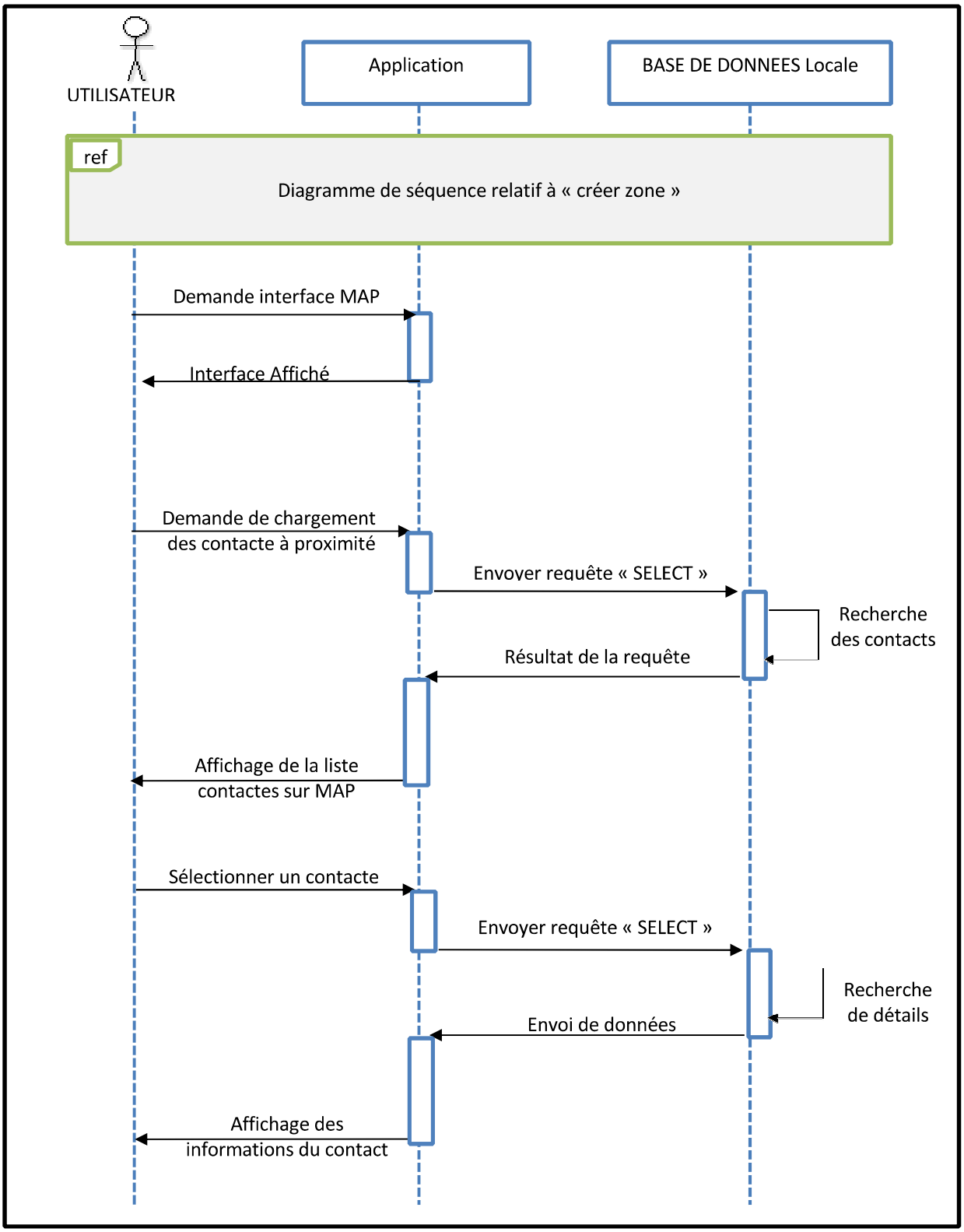


Figure 19 - Diagramme de séquence relatif à « localiser les contacts »

3.5 Conclusion

Lors de cette phase, nous avons présenté d'une manière globale et détaillée, le fonctionnement de notre application se basant principalement sur les diagrammes de cas d'utilisations et les diagrammes de séquence afin de faciliter l'implémentation.

Nous pouvons ainsi entamer la prochaine étape qui consiste à présenter la phase de réalisation.

Chapitre 4 Réalisation

4.1 Introduction

Après avoir détaillé la conception adaptée à notre application, nous allons consacrer le dernier chapitre de ce rapport à la partie réalisation. Pour cela nous allons présenter dans un premier lieu l'environnement matériel et logiciel de développement, par la suite, nous décrirons la phase d'implémentation en nous basant sur quelques interfaces.

4.2 4.1 Environnement de travail

4.2.1 Environnement matériel

Pour la réalisation de notre projet, nous avons utilisé un ordinateur DELL (Voir annexe) caractérisé par :

- Système d'exploitation : Windows 8.1.
- Processeur : Intel(R) Core(TM) i5-52000U CPU @ 2.20GHz.
- Mémoire vive : 8 Go.
- Disque Dur : 1 To.

Pour les différentes étapes de test, d'installation et le déploiement de l'application nous avons eu besoin des appareils mobiles supportant le système d'exploitation Android 2.3 (voir annexe) dont les caractéristiques sont les suivantes :

- Nom de l'appareil : Samsung Lite.
- Connexion : 3G, EDGE, GPRS.
- Bluetooth : Bluetooth 4.0.
- Port USB : USB 2.0.

- . Wi-Fi : 802.11a.
- GPS : GPS, Glonass.
- GPS : GPS, Glonass.
- Mémoire interne : 16 Go

4.2.2 Environnement logiciel

JAVA : « Java est un langage de programmation orienté objet, développé par Sun Microsystems. Il permet de créer des logiciels compatibles avec de nombreux systèmes d'exploitation (Windows, Linux, Macintosh, Solaris). Java donne aussi la possibilité de développer des programmes pour téléphones portables. » [11]

Android Studio : Android Studio est un environnement de développement intégré (IDE) pour le développement sur la plateforme Android. Il a été annoncé en mai 2013. Android est disponible librement sous la licence Apache 2.0. Basé sur le logiciel IDEA de JetBrains 'IntelliJ, Android Studio est conçu spécifiquement pour le développement Android. Il est disponible en téléchargement sur les systèmes d'exploitation ; Windows, Mac OS et Linux [13]. Android Studio permet principalement d'éditer les fichiers Java et les fichiers de configuration d'une application Android.

Il propose aussi des outils pour gérer le développement d'applications multilingues et permet de visualiser la mise en page des différents types et tailles d'écrans avec des résolutions variées simultanément. [12]

Java Development Kit (JDK) : « Le kit de Développent Java désigne un ensemble de bibliothèques logicielles de base du langage de programmation Java, ainsi que les outils avec lesquels le code Java peut être compilé, transformé en bytecode destiné à la machine virtuelle Java. » [13]

Android Software Development Kit (SDK) : Le SDK est un ensemble d'outils que met à disposition Google afin de nous permettre de développer des applications pour Android. Il est disponible pour Windows, MacOs X et linux et inclut des outils ainsi qu'un émulateur Android pour exécuter des applications.

SQLite est un système de base de données ou une bibliothèque proposant un moteur de base de données relationnelle. Il repose sur une écriture en C, un langage de programmation impératif, et sur une accessibilité via le langage SQL (Structured Query Language). [14]

ArgoUML est le principal outil de modélisation UML open source et inclut la prise en charge de tous les diagrammes UML 1.4 standard. Il fonctionne sur n'importe quelle plate-forme Java et est disponible en dix langues. [15]

4.2.2.1 Les technologies utilisées

JSON : JSON (JavaScript Object Notation – Notation Objet issue de JavaScript) est un format léger d'échange de données. Il est aisément analysable ou général par des machines. Il est basé sur un sous-ensemble du langage de programmation JavaScript. JSON est un format texte complètement indépendant de tout langage. [16]

Format JSON :

```
{
    "menu": {
        "id": "file",
        "value": "File",
        "popup": {
            "menuitem": [
                { "value": "New", "onclick": "CreateNewDoc()" },
                { "value": "Open", "onclick": "OpenDoc()" },
                { "value": "Close", "onclick": "CloseDoc()" }
            ]
        }
    }
}
```

API de géolocalisation de Google : La Google API est un kit de développement logiciel disponible de façon libre sur le web, qui offre la possibilité de créer de nouvelles applications qui utilisent la base de données des pages référencées et indexées par le géant Google et ceci grâce à des web services. [17]

OSMDroid : OSMDroid est un remplacement (presque) complet / gratuit de la classe MapView (v1 API) d'Android. Il inclut également un système de fournisseur de tuiles modulaires avec prise en charge de nombreuses sources de tuiles en ligne et hors ligne et un support de superposition avec des superpositions intégrées pour les icônes de tracé, l'emplacement de suivi et les formes de dessin. [18]

```
repositories {
        mavenCentral()
}
dependencies {
    compile 'org.osmdroid:osmdroid-android:<VERSION>'
}
```

4.2.3 Prénstation de l'application

Notre application mobile se base sur une communication P2Pentre un ensemble d'utilisateurs éditant une même carte géographique. Lors d'un cas d'urgence tel qu'un incendie ou un cas de catastrophe, un chef d'équipe dessine des zones sur le lieu de la catastrophe et affecte à chaque zone un ensemble de personnes qui vont saisir des données et décrire la situation dans différentes positions de la zone. Les utilisateurs éditant la même zone peuvent se communiquer à travers une interface de Chat.

Figure 20 -Interface de chargement de l'application

Dans ce qui suit, notre application sera présentée en exposant ses différentes interfaces mise en place avec succès. Les scénarios considèrent un cas d'une catastrophe naturelle où les utilisateurs sont des ambulanciers qui cherchent et sauvent les personnes dans le lieu de la catastrophe présenté dans l'application par une carte géographique.

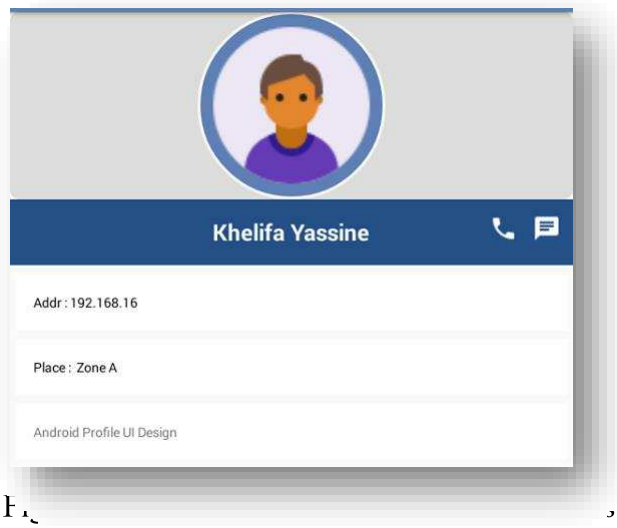
La figure 20 illustre l'Interface de chargement contenant le logo de l'application. Cette page dure trois secondes au maximum.

Lorsque l'application se lance pour la première fois, une interface comme présenté dans la figure 21 se lance et invite l'utilisateur à se connecter en introduisant son nom sans avoir besoin d'un compte au préalable.

Lorsque l'utilisateur clique sur le bouton "se connecter", un ID utilisateur se génère et ses données d'authentifications sont sauvegardées afin de pouvoir accéder directement pour les prochains lancements de l'application.

Comme le montre la figure 23, l'application est dotée d'un menu déroulant intuitif qui englobe toutes les fonctionnalités offertes par l'application.

Figure 21 - Interface de connexion d'un utilisateur



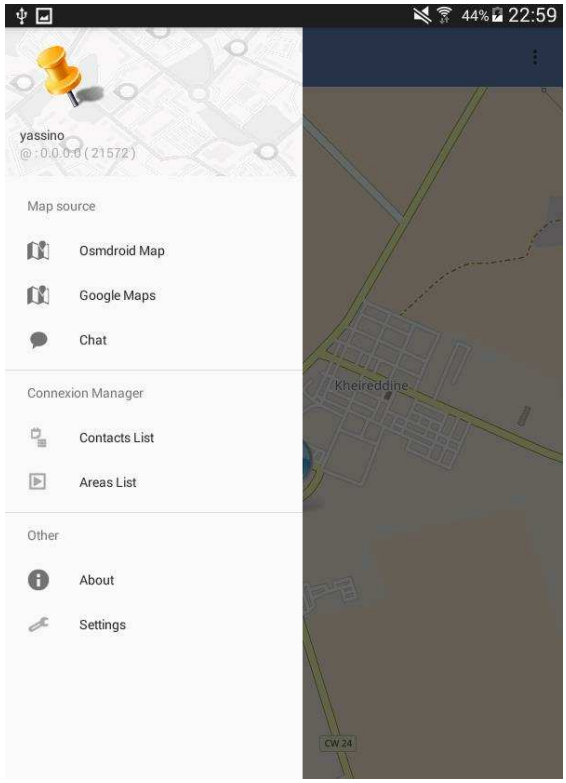


Figure 23-Interface menu principal de l'application

Lorsque l'utilisateur veut consulter les informations et l'état des autres utilisateurs connectés sur l'application il suffit de cliquer sur l'item « Map ». L'ensemble de ses informations s'affiche.

Chaque marker représente un utilisateur, il suffit de cliquer sur le marker pour savoir son nom et sa localisation comme indiqué sur la figure 22.

Pour se retrouver dans la carte graphique, l'utilisateur doit cliquer sur le bouton flottant en bas gauche de la fenêtre pour que sa position se rafraîchie.

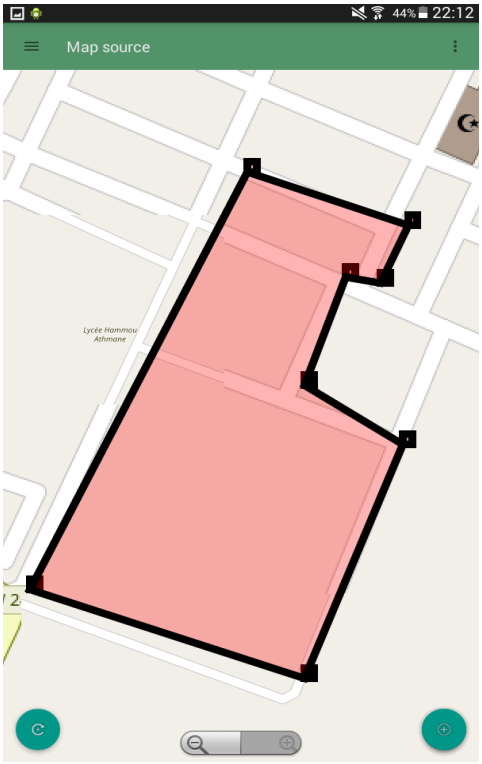


Figure 24 - interface de création d'une zone

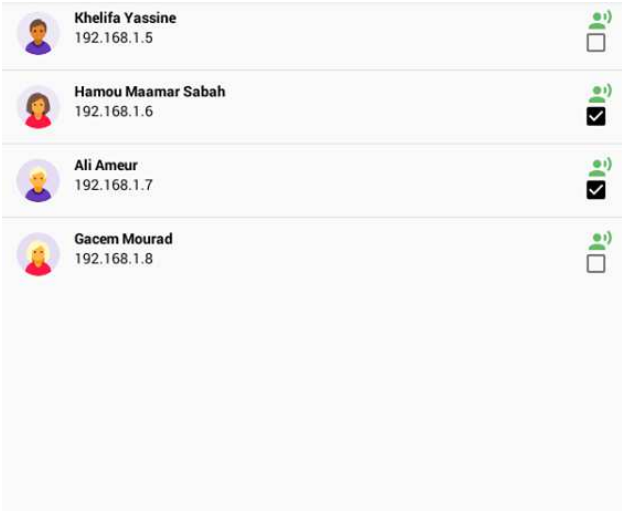


Figure 25 - interface de choix de des contacts

Le bouton flottant à droite en bas fait créer une nouvelle zone, lors du click au-dessus l'utilisateur dessine la zone sur laquelle il veut travailler comme indiqué à la figure 24 et clique suivant, un menu de contacts s'affiche et l'utilisateur est invité à choisir ceux qu'il veut ajouter à la zone comme le montre la figure 25. Une fois la liste est sélectionnée, la zone est créée et les utilisateurs choisis sont automatiquement entrés dans la zone.

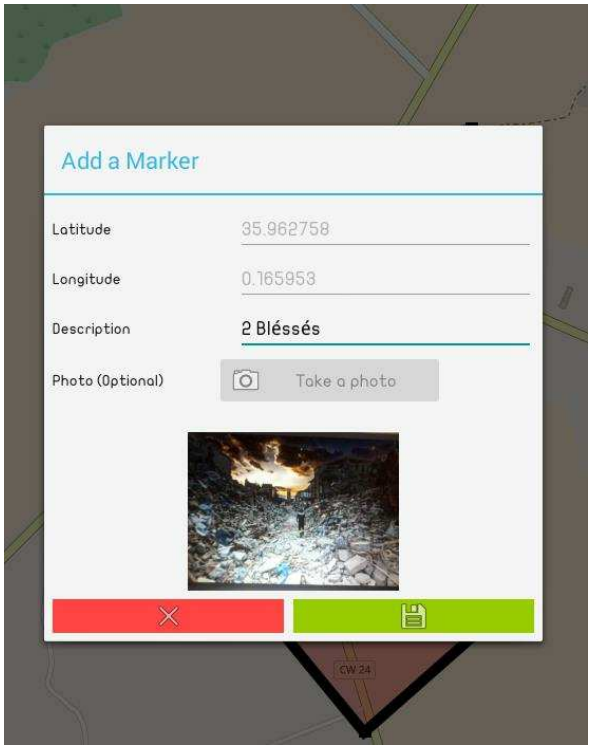


Figure 26 - dialogue d'ajout d'un marker

Une fois entré à la zone, l'utilisateur peut ajouter des markers via deux méthodes. Soit à travers le bouton flottant en bas à droite qui fait lancer le dialogue d'ajout montré dans la figure 26 avec la position courante de l'utilisateur. Soit à travers un long click sur la carte géographique et le marker sera ajouté avec les données de la location choisie.

Après la création du marker, il sera affiché sur la carte graphique et il suffit de cliquer au-dessus pour afficher les détails et lui donner la possibilité de le supprimer ou le modifier comme indiqué dans la figure 27. Lorsque l'utilisateur veut consulter ses contacts il suffit de cliquer sur l'item « Liste Contactes ». L'ensemble de ses contacts s'affiche dans une liste qui contient le nom, l'adresse et l'état de connectivité comme indiqué sur la figure 29 ci-dessous.

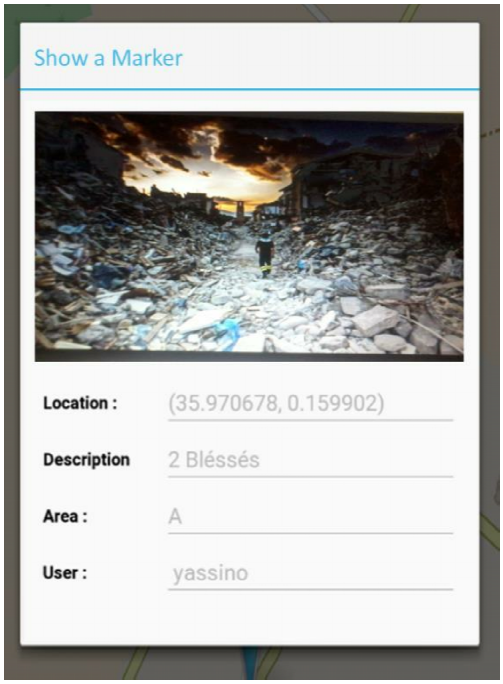


Figure 27 - Dialogue des détails du marker

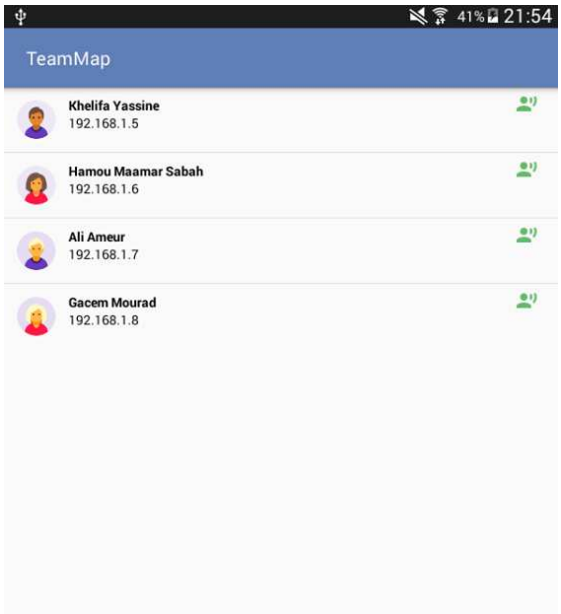


Figure 29 - Interface Liste contactes

Figure 28 - Interface de chat

Lorsque que l'utilisateur veut contacter un utilisateur il suffit de cliquer sur son item. Le chat s'ouvre comme le montre la figure 28.

4.3 Conclusion

Dans ce chapitre, nous avons essayé de présenter les techniques et les outils du développement de l'application réalisée. Nous avons illustré les fonctionnalités importantes de notre application en choisissant quelques interfaces graphiques et captures d'écran.

Conclusion Générale

Dans ce travail, nous avons présenté les systèmes d'éditions collaboratives des objets graphiques en temps réel tel que les Maps sur un système P2P répliqué. Nous avons aussi vu le modèle de cohérence Convergence-Causalité-Intention (CCI) permettant d'évaluer la préservation de causalité, d'intention et la convergence des répliques à tout moment dans le système et nous avons détaillé les approches utilisées pour l'édition collaborative telle que l'approche de CRDT et des transformées opérationnelles (TO).

Cette dernière permet de minimiser le nombre de versions d'objets pour s'adapter aux effets combinés d'opérations conflictuelles et compatibles et de résoudre également les problèmes de divergence intra-objet et inter-objet.

L'objectif de ce travail est de proposer un modèle collaboratif mobile à faible coût et flexible basé sur l'approche de Transformée Opérationnelle (TO), capable d'assurer la convergence des données et de préserver le travail produit simultanément par plusieurs utilisateurs face aux conflits, indépendamment de l'ordre d'exécution des mises-à-jour, dans les systèmes d'édition collaborative sur un objet graphique qui est une Map partagée via le réseau P2P en utilisant des appareil mobiles (Smart Phones). Cette application collaborative peut être utilisée dans différents domaines (urgence, tourisme, médical).

Comme perspectives, nous nous intéressons à paramétrer l'entourage de la carte géographique et minimiser l'envoi des images dans le réseau. Nous pensons aussi à faire fonctionner ce modèle en une architecture hybride pour permettre l'édition collaborative via internet.

Bibliographie

[1] Mehdi Ahmed-Nacer, Pascal Urso, Claudia-Lavinia Ignat, Gérald Oster : *Évaluation de l'occupation mémoire des CRDTs pour l'édition collaborative temps-réel mobile*, UbiMob French-speaking Conference on Mobility and Ubiquity Computing, France, 2012

[2] C.M. Neuwirth, R. Chandhok, D.S. Kaufer, P. Erion, J. Morris, et D. Miller : *Fléxibilité dans les Systèmes d'édition collaborative, Conférence sur le travail collaboratif*, pages 147—134, Toronto, Italie, Novembre, 1992.

[3] S. Greenberg : *Le travail coopératif assisté par ordinateur,* Academic Press, Londre, 1991.

[4] David Chen : *Maintenance de cohérence dans les systèmes d'édition de graphiques collaboratifs*, Thèse de doctorat.Université de Griffith,2001

[5] Xavier Défago, André Schiper, et Péter Urbàn : *Total order broadcast and multicast algorithms* : Taxonomy and survey, ACM Computing Surveys, 36(4), pages 372–421, 2004.

[6] Abdessamad IMINE : *Conception Formelle d'Algorithmes de Réplication Optimiste Vers l'Edition Collaborative dans les Réseaux P2P*, Thèse de doctorat en informatique, Nancy : université Henri Poincaré, France, 2006.

[7] C.A. Ellis et S.J. Gibbs : *Concurrency control in groupware systems.* ln proc.of ACMSIGMOD, Conferénce en Gestion de données, pages 399—407, Mai, 1989.

[8] Gérald Oster : *Réplication optimiste et cohérence des données dans les environnements collaboratifs répartis*, 2005.

[9] Pascal Molli, Gerald Oster, Hala Skaf-Molli, Abdessamad Imine : *Safe Generic Data Synchronizer*, LORIA, France, 2003.

[10] R. Elmasri et S. B. Navathe : *Les fondamentaux des systèmes de bases de données*,à benjamin/cummings, 1989.

[11] futura-sciences : http://www.futura-sciences.com/tech/definitions/internet-java-485/

[12] zdnet: Android Studio, une version 1.0 pour l'IDE de Google. http://www.zdnet.fr/actualites/android-studio-une-version-10-pour-l-ide-de-google-39811025.html

[13] Comment çamarche : Java – Introduction, http://www.commentcamarche.net/contents/559-java-introduction

[14] journaldunet : https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203607-sqlite-definition/

[15] ArgoUML: http://argouml.tigris.org/

[16] JSON : Présentation de JSON,http://www.json.org/jsonfr.html

[17] webrankinfo : Olivier, D : Les API de Google Développer un outil avec la Google API http://www.webrankinfo.com/dossiers/apigoogle/api-soap#gref

[18] osmdroid sur github : http://osmdroid.github.io/osmdroid/