

Table des matières

Table des matières	i
1 Introduction	1
1.1 Contexte	2
1.1.1 La collaboration	2
1.1.2 Les environnements virtuels collaboratifs 3D	5
1.1.3 Les systèmes d'édition collaboratifs	7
1.1.4 Les architectures orientées événements pour la collaboration	8
1.2 Problématique	12
1.3 Contributions	14
1.3.1 Contributions théoriques	14
1.3.2 Contributions pratiques	15
1.4 Organisation du manuscrit	15
2 État de l'art	16
2.1 Modélisation 3D collaborative sur le web	17
2.1.1 Collaboration et concurrence en 3D	17
2.1.2 ... sur le web : HTML5 et au-delà	19
2.1.3 Web 3D : deux approches	21
2.2 Communication en temps-réel	24
2.2.1 Fiable versus Non Fiable	26
2.2.2 Ordonné versus Désordonné	27
2.2.3 Les principaux protocoles de transport	28
2.2.4 Web et P2P : WebRTC	30
2.2.5 Quel protocole dans quel EVC3D ?	33
2.3 Les systèmes distribués orientés événements pour la collaboration	35
2.3.1 Introduction	35
2.3.2 Systèmes Publish-Subscribe	36
2.3.3 Domain Driven Design	39
2.3.4 Command Query Responsability Segregation	42
2.3.5 Event Sourcing	43
2.4 Conclusion	48
3 Contributions scientifiques	50
3.1 Introduction	51
3.2 Modèle événementiel pour l'intégration du domaine 3D dans les EVC	52
3.2.1 Modèle général	54

3.2.2	Adaptation des patrons Event Sourcing (ES) et Command Query Responsability Segregation (CQRS)	56
3.2.3	Cohérence éventuelle en CQRS	60
3.2.4	Potentielles applications et autres utilisations	61
3.2.5	Bilan	62
3.3	Architecture de communication hybride	63
3.3.1	Éléments constitutifs d'un pair	66
3.3.2	Architecture hybride « orientée états »	69
3.3.3	De l'état à l'événement	74
3.3.4	Architecture de communication hybride « orientée événements »	75
3.3.5	Comparaison des deux architectures	80
3.4	Conclusion du chapitre	80
4	Implantation	82
4.1	Introduction	83
4.2	3DEvent : Plateforme web de manipulation collaborative d'objets 3D	83
4.2.1	Éditeur 3DEvent	83
4.2.2	Interface utilisateur orientée tâche	84
4.2.3	Flexibilité de la visualisation	90
4.2.4	Bilan	90
4.3	Intergiciel P2P pour l'échange de données 3D pour le paradigme événementiel	91
4.3.1	Données d'échange	92
4.3.2	Synchronisation des données	93
4.4	Résumé des choix techniques	95
4.5	Bilan	97
5	Expérimentations	98
5.1	Cas d'étude : Assemblage collaboratif d'objets 3D dans un environnement web	99
5.2	Expérimentation 1 : preuve de faisabilité	100
5.2.1	Présentation de l'expérimentation 1	100
5.2.2	Résultats et discussion	102
5.2.3	Conclusion de l'expérimentation 1	103
5.3	Expérimentation 2 : intégration du framework événementiel	104
5.3.1	Présentation de l'expérimentation 2	104
5.3.2	Résultats et discussion	106
5.3.3	Conclusion de l'expérimentation 2	108
5.4	Comparaison entre l'expérimentation 1 et l'expérimentation 2	109
6	Conclusion	115
A	Ressources pour l'implantation et les expérimentations	118
A.1	Description des événements par agrégat dans 3DEvent	118
A.2	Messages réseaux pour la synchronisation des Event Stores	120
A.3	Experimentation 1 materials	121
A.3.1	Description des modèles utilisés	121
A.3.2	Questionnaire pour l'enquête utilisateur	121

A.4 Expérimentation 2	121
A.4.1 Description des modèles utilisés	121
A.4.2 Questionnaire pour l'enquête utilisateur	121

Chapitre 1

Introduction

Sommaire

1.1	Contexte	2
1.1.1	La collaboration	2
1.1.2	Les environnements virtuels collaboratifs 3D	5
1.1.3	Les systèmes d'édition collaboratifs	7
1.1.4	Les architectures orientées événements pour la collaboration	8
1.2	Problématique	12
1.3	Contributions	14
1.3.1	Contributions théoriques	14
1.3.2	Contributions pratiques	15
1.4	Organisation du manuscrit	15

1.1 Contexte

Le besoin de collaboration et de manipulation en temps réel d'objets en **trois dimensions (3D)** ainsi que leur contrôle de version a été très tôt une raison de la dissémination des outils pour la **Conception Assistée par Ordinateur (CAO)**. Tandis que l'ingénierie et la visualisation scientifique ont produit des quantités de données massives dans des domaines tels que la conception architecturale, l'industrie des jeux ou encore l'impression **3D**, un besoin croissant de maintenir, visualiser et manipuler de larges scènes **3D** polygonales pouvant être éditées par de multiples utilisateurs de manière concurrente se fait sentir [Chandrasegaran *et al.*, 2013, Wu *et al.*, 2014]. Or la quantité et la taille des données étant toujours croissante, il devient de plus en plus difficile de partager les modèles, particulièrement avec les utilisateurs n'ayant pas accès aux derniers logiciels et matériels graphiques. En conséquence, le développement de plateformes légères déployées sur le web pour répondre à ces besoins est en forte progression. Dans un **Environnement Virtuel 3D (EV 3D)**, la simulation virtuelle et simulation distribuée d'un environnement **3D**, un grand nombre d'utilisateurs situés à différents endroits géographiques peuvent interagir les uns avec les autres en temps réel. Un **Environnement Virtuel Collaboratif 3D (EVC 3D)** insiste sur l'aspect collaboratif des interactions qui se produisent entre les utilisateurs. Cette distorsion de l'espace physique et temporel impose aux **EVCs 3D** des mécanismes de communication rapides, conservant un environnement cohérent des données partagées durant la collaboration. L'utilisation d'un client comme medium pour accéder à l'**EVC 3D** est requise pour envoyer des demandes au serveur. Le client peut être manipulé par un utilisateur ou un agent logiciel (bot). Un **EVC 3D** se compose également d'un protocole de communication qui permet aux différents clients d'échanger les mises à jours correspondant aux modifications effectuées dans l'espace **3D** virtuel partagé. La distribution des données devient alors un enjeu majeur dans ce type d'application en terme de temps, de sécurité et de fiabilité.

1.1.1 La collaboration

La collaboration est souvent définie comme un processus récursif¹ où au moins deux acteurs (personnes ou organisations) travaillent ensemble à la croisée de buts communs en partageant leurs connaissances pour apprendre et bâtir un consensus. La collaboration permet l'émergence de conceptions partagées dans la réalisation de visions partagées

1. Le « principe de boucle récursive » se retrouve dans le concept de la pensée complexe. Edgar Morin explique qu'« un processus récursif est un processus où les produits et les effets sont en même temps causes et producteurs de ce qui les produit » [Morin, 1990a, p. 100].

dans des environnements et des systèmes complexes. Les imbrications de chaque domaine et la transdisciplinarité sont acceptés comme dans le concept de pensée complexe. Dans sa définition de la complexité, Edgar Morin fait d'ailleurs référence à l'étymologie latine « *complexus* » qui signifie « ce qui est tissé ensemble » [Morin, 1990b]. La plupart des collaborations requièrent un élément dirigeant qui peut prendre une forme sociale (personne) au sein d'un groupe décentralisé et égalitaire (horizontal). L'élément dirigeant va également souvent aider à trouver des consensus (exemple : la disponibilité des ressources). Une équipe travaillant de manière collaborative a la possibilité de concentrer plus de ressources, de reconnaissances et de récompenses lors d'une compétition comportant des ressources finies. La collaboration est aussi présente dans la recherche de buts opposés mettant en avant la notion de collaboration contradictoire (en opposition avec la collaboration constructive) ; la négociation et la compétition peuvent également faire partie du terrain de la collaboration.

Une application collaborative peut aussi intégrer les notions de coordination et de coopération :

- La **coordination** se base sur le principe d'harmonisation des tâches, des rôles et du calendrier dans des systèmes et environnements simples.
- La **coopération** permet de résoudre des problèmes dans des systèmes et environnements complexes dans lesquels les participants auraient été incapables (temps, espace, connaissance, matériel) d'accomplir le travail seuls.

Dans les années 2000, deux classifications ont été retenues concernant la collaboration. La première classification, décrite en 2007 par Gotta [Gotta, 2007], propose un modèle segmentant la collaboration de manière structurée en quatre catégories : de la plus dirigée à la plus volontaire, en passant par l'hybride.

- *Collaboration centrée processus*. Les conditions requises du processus nécessitent l'engagement de l'utilisateur, qui doit, de par son rôle ou sa responsabilité, diriger ses efforts dans la collaboration avec les autres. Cette stratégie se concentre sur les activités de manipulation collaborative 3D plutôt que sur leur contexte organisationnel afin de favoriser la synergie autour de la réussite d'un processus. Par exemple, pour la création et l'utilisation d'un modèle 3D dans un **Business Information Modeling (BIM)**, il s'agit de favoriser la prise de décision sur un projet et de communiquer à ce propos.
- *Collaboration centrée activité*. Les activités partagées créent un sentiment de co-dépendance qui induit la collaboration entre les membres. La co-dépendance prend l'avantage sur l'intérêt personnel comme motivation à collaborer. Le groupe a besoin

de chacun pour que l'objectif soit considéré comme réalisé. L'intérêt personnel ou l'allégeance à l'esprit d'équipe peut aussi promouvoir la collaboration. Par exemple, la visualisation collaborative des activités des différents contributeurs dans l'[EVC 3D](#) permet à chacun de rendre compte de ses réalisations. Ce type de collaboration repose beaucoup sur l'ergonomie de l'activité qui insiste sur la différence entre le travail prescrit et le travail réel : la tâche et l'activité.

- *Collaboration centrée communauté.* La participation de la communauté à la collaboration induit la contribution. En effet, les interactions professionnelles ou sociales peuvent encourager ou persuader les utilisateurs de partager leurs informations ou connaissances (exemple : les logiciels *open-source*)
- *Collaboration centrée réseau.* Les connexions réseau favorisent la coopération réciproque. Dans le but de récupérer des avis ou du savoir-faire externe, un utilisateur peut faire appel à son réseau social pour compléter une autre interaction collaborative. Souvent utilisée dans le cadre d'urgences écologiques ou sanitaires (exemple : contributions OpenStreetMap lors d'ouragans ou de tsunamis), la collaboration centrée réseau est très présente dans des situations où l'expertise est fortement valorisée comme dans les [BIM](#) ou la visualisation scientifique. Les contributeurs peuvent être intégrés en fonction des besoins des utilisateurs déjà présents sur le projet.

Dans le cadre de cette thèse, en se référant à cette première classification, les aspects centrés sur les activités sont mis en avant. En effet, la collaboration portant sur la modélisation [3D](#) attend un résultat porté sur l'activité de conception. Celle-ci nécessite l'implication de personnes dotées de différentes compétences / connaissances qui doivent s'entraider pour mettre en commun des objets [3D](#) et réaliser leur objectif.

Une seconde classification proposée par Callahan et al. [Callahan et al., 2008] s'intéresse au triplé « collaboration par équipe », « collaboration communautaire » et « collaboration en réseau ». En contraste avec la précédente classification qui se concentre sur les équipes et une collaboration formelle et structurée, celle-ci offre davantage d'ouverture :

- *Collaboration par équipe.* Dans une équipe, tous les membres se connaissent. Il y a une interdépendance claire des tâches à effectuer où la réciprocité est attendue, avec un échéancier et des objectifs explicites. Pour réaliser son but, l'équipe doit réaliser les tâches dans un temps imparti. La collaboration par équipe suggère que les membres coopèrent sur un pied d'égalité (bien qu'il y ait souvent un chef) recevant une reconnaissance égale.
- *Collaboration communautaire.* L'objectif de ce type de collaboration est plus orienté sur la possibilité d'apprendre que sur la tâche elle-même, même si les centres d'in-

térêt sont partagés par la communauté. Les utilisateurs sont là pour partager et construire la connaissance davantage que pour compléter un projet. Les membres vont aller voir leur communauté pour demander de l'aide sur un problème ou un avis et rapporter la solution à implémenter dans leur équipe. L'adhésion peut être limitée et explicite, mais les périodes de temps sont souvent ouvertes. Les membres sont considérés comme égaux, bien que les plus expérimentés puissent avoir des statuts privilégiés. La réciprocité dans la communauté est un facteur important pour que cela fonctionne.

- *Collaboration par réseautage.* La collaboration en réseau est une surcouche de la collaboration traditionnellement centrée sur la relation d'une équipe ou d'une communauté. Elle s'appuie sur une action individuelle et un intérêt personnel qui resurgissent ensuite sur le réseau, sous la forme de personnes qui contribuent ou cherchent quelque chose à partir du réseau. L'adhésion et les périodes sont ouvertes et non limitées. Il n'y a pas de rôle explicite. Les membres ne se connaissent pas forcément. Le pouvoir est distribué. Cette forme de collaboration est dirigée par l'avènement des réseaux sociaux, d'un accès omniprésent à internet et de la capacité de se connecter avec divers individus malgré la distance.

Cette thèse, en se référant à cette seconde classification, se concentre sur la collaboration par équipe. La conception d'un objet **3D**, et ses différentes phases de modélisation, constitue une problématique nécessitant l'apport de plusieurs intervenants avec leurs capacités propres et travaillant de concert à la réalisation d'un objectif commun dans un temps imparti (exemple : revue de projet). Là où la coopération et l'effort conjoint pour réaliser un objectif sont nécessaires, le facteur temps reste un élément important à prendre en compte pour évaluer la productivité d'une session collaborative.

Le travail dans un **EVC 3D** facilite la compréhension de certaines problématiques liées à l'espace **3D**; c'est également un point de rencontre et d'échanges entre contributeurs sur le court terme et le long terme. Le croisement de ces deux dimensions, spatiale et temporelle, implique une multiplication des points de vue et donc des données à traiter sur le problème lors de la collaboration.

1.1.2 Les environnements virtuels collaboratifs **3D**

Un **EVC 3D** est un environnement virtuel **3D** où plusieurs utilisateurs locaux ou à distance peuvent se rejoindre et partager une expérience collaborative interactive en **3D**. La principale caractéristique d'un **EVC 3D** est la simulation immersive et interactive **3D** d'environnements virtuels, tels que les jeux sérieux ou multijoueurs. **EVC 3D** permet à

plusieurs utilisateurs d’interagir les uns avec les autres en temps (quasi) réel, même s’ils sont situés dans des lieux différents. D’autres fonctionnalités sont proposées par ce type de plateformes comme le partage d’espace, de présence et du temps. Selon Singhal et Zyda [Singhal et Zyda, 1999], **EVC 3D** est constitué de quatre composants principaux : (i) un moteur graphique pour l’affichage ; (ii) des appareils de contrôle et de communication ; (iii) un système de traitement ; (iv) et un réseau de transmission des données.

Ce type d’environnement nécessite également d’impliquer l’utilisateur dans une boucle « action / perception » pour lui permettre prendre conscience que ses actions contribuent à la modification de l’environnement virtuel. Les interactions classiques se font par le biais d’appareils dédiés. Il existe trois catégories d’interactions selon Chris Hand [Hand, 1997] : la navigation (interaction avec le point de vue de l’utilisateur) ; la manipulation des objets virtuels issus de l’environnement virtuel (sélection d’objet, manipulation d’objet) ; et le contrôle de l’application (interaction avec une interface **3D** pour modifier des paramètres de l’environnement virtuel).

La manipulation d’objets figure parmi les interactions les plus fondamentales dans la **CAO**, le **BIM** ou le **Product Lifecycle Management (PLM)**. La manipulation collaborative d’objets virtuels par plusieurs utilisateurs fait ainsi partie des nouveaux besoins liés aux développements de ces activités. Elle s’avère indispensable dans plusieurs types d’applications comme le prototypage virtuel, les simulations d’entraînement ou la simulation d’assemblage et de maintenance. L’expérience de modélisation **CAO** collaborative peut être accompagnée de mécanismes ludiques pour intégrer une capture temps réel de la connaissance [Kosmadoudi *et al.*, 2013]. Ces lieux virtuels sont l’occasion pour les utilisateurs de participer de manière naturelle et efficace à la création et à la vie de l’objet manipulé dans l’environnement virtuel sans danger et à bas coût. Une autre utilisation des **EVC 3D** sert la navigation virtuelle, c’est-à-dire les visites collaboratives (musées, héritage culturel, les revues de projets architecturaux / urbains, ou encore les jeux collaboratifs (courses, simulateur de jeux collectifs). Les **EVC 3D** permettent non seulement aux utilisateurs de communiquer de manière distante, mais aussi de partager des interactions dans le monde virtuel. Ces interactions peuvent s’appliquer à différents objets, différentes parties du même objet, ou encore sur la même partie (en même temps) d’un objet virtuel partagé. Plusieurs problèmes, liés aux domaines des systèmes distribués (et des protocoles réseaux) et d’**IHM**, doivent être résolus pour concevoir un **EVC 3D** fiable, ergonomique et en temps réel.

1.1.3 Les systèmes d'édition collaboratifs

En 1989, Ellis et Gibbs proposent la définition d'un *groupware*, terme généralement traduit par collecticiel : « *computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment.* » [Ellis et Gibbs, 1989]. Une application collaborative est une application qui accompagne un groupe de participants dans la réalisation d'une tâche ou d'un objectif en fournissant une interface vers un environnement partagé. Cette définition, assez large, englobe un ensemble très hétérogène d'applications permettant le travail collaboratif.

Vidot propose de distinguer les différentes catégories de collecticiels en fonction de leur rapport au temps et à l'espace [Vidot, 2002]. L'aspect temporel se réfère à la synchronicité des interactions. Une synchronicité élevée implique que les actions des utilisateurs sont visibles en temps réel (action synchrone). Au contraire, une synchronicité faible, implique qu'un temps notable s'écoule entre l'action d'un utilisateur et sa visibilité chez les autres (action asynchrone). L'aspect spatial s'intéresse à la répartition géographique des utilisateurs. Les environnements sont dits répartis lorsque les utilisateurs sont physiquement distants. Cette thèse concerne les applications collaboratives appartenant à la catégorie des applications synchrones réparties.

Un modèle d'édition collaborative en temps réel permet à plusieurs utilisateurs de visualiser et d'éditer de manière simultanée le même document (texte, image, 3D) depuis plusieurs sites connectés par un réseau de communication. Un modèle d'édition collaborative se compose d'un nombre connu ou inconnu de répliques. Une réplique est une copie du document partagé modifiable à n'importe quel instant. L'exécution d'une opération de modification se fait localement sur la réplique, qui la diffuse ensuite à l'ensemble des répliques. La notification de la modification, contenant l'opération de modification, est communiquée de manière asynchrone. Les répliques distantes qui reçoivent cette notification exécutent alors l'opération localement. L'une des difficultés principales de ces systèmes est d'en maintenir la cohérence. Il existe différents facteurs pouvant pousser un système à ne plus être cohérent :

- **La divergence.** Les opérations peuvent arriver sur plusieurs sites dans différents ordres, provoquant des résultats dissemblables sur chaque réplique participant à l'édition collaborative - à moins que les opérations ne soient commutatives (ce qui est rarement le cas). Dans les cas d'application où la cohérence du résultat final est nécessaire, la divergence doit être évitée. C'est possible grâce à l'utilisation d'un protocole de sérialisation.

- **La violation de la causalité.** Le fait que la latence dans une communication soit intrinsèquement non-déterministe peut conduire le système à une violation de la causalité : les opérations peuvent arriver et être exécutées dans un ordre qui ne respecte pas l'ordre causal.
- **La violation de l'intention.** Suite à la génération d'opérations concurrentes, l'effet d'une opération au moment de son exécution peut être différent de l'intention de cette opération au moment de sa génération.

Ces trois facteurs d'incohérence sont indépendants, notamment en ce qui concerne la violation de l'intention et le problème de divergence qui sont deux problèmes d'incohérence de différente nature : le premier facteur peut toujours être résolu en utilisant un protocole de sérialisation alors que le dernier ne peut utiliser un protocole de sérialisation fixé si les opérations sont exécutées sous leurs formes originelles [Sun et al., 1997]. Le modèle **Causality, Convergence, Intention (CCI)** proposé par Sun et al. [Sun et al., 1998] propose un modèle de cohérence pour les systèmes d'édition collaborative qui aborde ces problèmes afin d'assurer « la préservation de la convergence, de la causalité et la préservation de l'intention » [Sun et al., 1998]. Une majorité des travaux qui se sont intéressés aux propriétés **CCI** portent sur des documents textuels [Weiss et al., 2010].

Cette thèse s'intéresse précisément à l'édition collaborative d'espace de travail en 3D. Sont considérées ici les opérations d'ajout et de suppression d'éléments **3D** dans un espace partagé, ainsi que les modifications de haut niveau sur ces éléments **3D** telles que la translation, la rotation et l'homothétie.

1.1.4 Les architectures orientées événements pour la collaboration

Un événement est un élément omniprésent de la vie. Le terme *événement* existe dans presque tous les champs en science, avec différents sens. Le but de cette section est de répertorier les différentes notions d'événements. Pour cela, différentes descriptions du terme sont proposées ainsi que leur notion liée à la littérature informatique.

Dans la littérature, une variété de définitions du terme événement existe. Habituellement, un événement est considéré comme quelque chose qui « se produit », en particulier quelque chose d'habituel ou d'important. Cependant la plupart des travaux de la littérature évitent de définir le sens précis des événements en n'indiquant pas le champ d'application dans lequel ils sont utilisés. Il existe trois entrées pour le terme événement dans le dictionnaire. La première entrée se réfère à la physique (dans le cadre de la théorie de la relativité), où un événement est « un phénomène considéré comme localisé et ins-

tantané, survenant en un point et un instant bien déterminés ». La seconde se réfère à la théorie des probabilités, indiquant qu'un événement est « la partie d'un univers Ω réalisée quand l'une des éventualités la composant se réalise ». La troisième définition se rapporte à la psychologie impliquant « tout ce qui est capable de modifier la réalité interne d'un sujet (fait extérieur, représentation, etc.) ». Cette dernière définition est très proche de ce que l'on retrouve en informatique, comme les changements d'état ou les actions entraînant certaines conséquences. Dans une application, il peut être important de s'intéresser au déplacement d'un objet de quelques unités (*object moved event*) ou d'apprendre qu'un utilisateur est passé de déconnecté à connecté (*status changed*). On s'applique donc à identifier ce qui modifie l'état intrinsèque d'un objet et sa représentation. Bien que ces différentes descriptions permettent d'avoir une compréhension générale du terme événement, chaque sous-discipline de l'informatique comprend ses propres associations.

L'implantation d'une architecture orientée événements (de l'anglais **Event Driven Architecture (EDA)**) nécessite l'instanciation d'une architecture abstraite, le positionnement des composants sur des machines, ainsi que des protocoles pour subvenir à l'interaction, en utilisant des technologies et des produits spécifiques. Une telle instance est appelée architecture système. De ce fait, les architectures de systèmes distribués orientés événements doivent répondre aux exigences des utilisateurs et aux problèmes liés à la nature des plateformes et applications. Le passage à l'échelle (nombre d'utilisateurs, ressources distribuées sur de grandes zones géographiques) génère un grand nombre d'événements qui doivent être traités de façon efficace.

Les systèmes complexes distribués sont construits à partir de collections couplées de manière dite « lâche » (*loosely coupled*)², technologiquement neutre et indépendante de la localisation des services. Le développement d'applications dirigées par les événements est un challenge tripartite : la production d'événements, le traitement des événements et la consommation des événements [Chandy et al., 2011]. Cristea et al. [Cristea et al., 2011] présentent un aperçu des architectures distribuées pour les systèmes orientés événements. Cette approche est utilisée dans de nombreuses applications réactives. Souvent appliquées à la finance ou aux systèmes logistiques, de telles solutions peuvent également intégrer les besoins de plateformes distribuées à grande échelle, comme les applications web et le travail collaboratif.

2. Un couplage lâche indique que les composants échangent peu d'informations. Contrairement à un couplage fort, où les composants ont une faible indépendance fonctionnelle et sont donc peu réutilisables, le couplage faible s'appuie sur l'établissement d'un protocole d'échange faisant le moins d'a priori sur les composants. Cela permet de fixer un cadre d'interaction entre les composants.

Sensibilisation et perception du groupe

Le travail collaboratif peut également être supporté par des modèles orientés événements prenant en compte la sensibilisation au groupe (*awareness*)³ dans des activités. Ces systèmes sont présents dans la littérature depuis les prémisses de la technologie web [Bentley et Wakefield, 1997, Steinfield *et al.*, 1999, You et Pekkola, 2001]. Ces propositions ont commencé par approcher la sensibilisation à l'espace de travail dans le but d'informer les utilisateurs des changements se produisant dans l'espace de travail partagé. Des travaux plus récents se sont concentrés sur des nouveaux paradigmes comme les systèmes **pair à pair (P2P)** pour proposer des *groupware* décentralisés ubiquitaires et sensibles à l'environnement. La sensibilisation au sein du groupe n'est pas seulement utilisée pour notifier les utilisateurs ; son but est également d'aider dans les processus de groupes pour éviter les problèmes. Ces derniers peuvent prendre différentes formes comme : l'inefficacité due à une information limitée ou la faiblesse d'un système de communication ; la disponibilité d'informations superflues ; la difficulté d'extraire l'information pour surveiller ou faire des rapports ; l'utilisation des données issues des ressources produites par le groupe pour améliorer sa perception des données disponibles. Par exemple, Xhafa et Poulovassilis [Xhafa et Poulovassilis, 2010] exposent une approche distribuée basée événements pour gérer des collecticiels (*groupware*) en **P2P**. Leur méthode permet de développer des applications de collecticiels spécifiques sur les opérations et les services primitifs liés à la sensibilisation. Ces derniers sont directement implémentés dans l'intericiel **P2P**. Le modèle propose différentes approches dépendant de la plateforme (web ou mobile) avec une granularité différente de l'information d'*awareness* dont :

La sensibilisation distribuée A l'inverse d'une approche centralisée, où l'information de sensibilisation est gérée par un serveur central, dans un système décentralisé, l'information sur laquelle la sensibilisation est construite est répartie sur les sites des différents pairs. Dans les approches basées serveur, le traitement des événements est effectué sur le serveur qui stocke les événements ainsi que la base de données contenant l'historique et fournit l'interface de requête pour extraire les informations d'intérêt. Dans le collecticiel **P2P**, le stockage, le traitement et les requêtes d'événements doivent être faits de manière répartie.

La sensibilisation à la dynamicité des événements Les systèmes **P2P** sont dynamiques par nature (*join / leave* des pairs). La fonctionnalité de sensibilisation doit prendre en compte cette dynamicité. En effet, la synchronisation et la cohérence de

3. La traduction d'*awareness* par sensibilisation en français est un peu réductrice car le terme anglais incorpore la perception de l'environnement ambiant, être conscient de ce qui se passe autour

l'information sur les différents sites sont cruciales et plus difficiles à mettre en place que dans un système client-serveur. La sensibilisation sous contrainte de dynamicité du réseau doit intégrer des mécanismes de propagation et de réPLICATION fournissant le contenu au groupe.

La généricité des événements Un système basé événements manipule plusieurs types d'événements. C'est pourquoi il est nécessaire que les événements soient le plus génériques possibles pour faciliter la construction de requêtes à travers un système lâchement couplé.

Les mécanismes à empreinte mémoire réduite L'utilisation de mécanismes à empreinte mémoire réduite est indispensable dans le but de réduire la surcharge causée par la génération d'événements, le traitement des événements et les notifications, ainsi que pour permettre le support de la sensibilisation par des pairs qui possèdent des capacités limitées.

Brown et al. [Brown et al., 2003] proposent un mécanisme de distribution de données basé événements. Le *Battlefield Augmented Reality System* (BARS) se situe dans le contexte de la collaboration sur mobile en réalité augmentée et environnement virtuel. Cet environnement a besoin de conserver une information cohérente au cours du temps, qui plus est de rendre compte de la situation (*situation awareness*) et de permettre la coordination d'équipe sur mobile entre les utilisateurs. Ils définissent *situation awareness* comme le fait que « chaque utilisateur doit obtenir une meilleure compréhension de l'environnement ». Pour cela, ils se placent dans un cadre où il est nécessaire de gérer plusieurs utilisateurs avec des connexions différentes (bas débit et haut débit) avec des connexions au réseau qui ne sont pas fiables et une réPLICATION partielle des données pour minimiser les événements non voulus. L'analyse des applications utilisant des systèmes orientés événements explique qu'ils sont obligés de faire des choix spécifiques au métier lié à l'application [Hinze et al., 2009]. Par exemple, dans les applications de jeux vidéo, la distribution des événements est souvent liée à un mécanisme **Publish-Subscribe (Pub / Sub)** centralisé qui modifie les souscriptions au fur et à mesure que les joueurs se déplacent dans l'espace. De plus, les événements dans un jeu doivent être protégés contre l'altération pour éviter la triche ou la dissémination d'événements faux. Cela s'applique également aux données d'un **EVC 3D** destiné à un usage industriel (données confidentielles ou critiques).

Intégration des règles métiers

Comme les bureaux d'études en ingénierie et en architecture travaillent sur des projets (visualisation **CAO**, **BIM**, gestion et arrangement d'espaces architecturaux), la collabo-

ration de professionnels venant de milieux différents avec des compétences et connaissances variées doit pouvoir s'accomplir autour d'un outil qui connaît leur langage, leurs contraintes - leur expertise. En effet, les modifications de modèles en 3D doivent être revues par des gestionnaires de projet, des clients et les intervenants impliqués qui peuvent à leur tour suggérer des modifications sur la conception. Tout cela doit se faire en accord avec des contraintes métiers transparentes. Les règles métier doivent donc apparaître dès la conception pour être intégrées tôt dans la modélisation 3D.

Les architectures orientées événements sont bien adaptées pour intégrer dans la description des événements plusieurs aspects liés au métier. Cette sémantique porte avec elle les connaissances des manipulations expertes. Un aspect majeur de cette thèse repose sur l'intégration du métier dans le processus de la manipulation et de la visualisation des objets 3D.

À l'ajout d'information sémantique sur les données, les EDAs enrichissent les applications de la possibilité d'implantation d'outils de surveillance de flux métiers pour l'observation en temps réel ou l'analyse a posteriori de ces données pour fixer des objectifs ou repérer des problèmes de performance dans la collaboration par exemple.

1.2 Problématique

Cette thèse se situe à l'interface de trois champs de recherches (Figure 1.1). Le premier concerne les environnements de modélisation 3D. Souvent commerciaux (Clara.io, OnShape, Verold Studio), les modeleurs utilisent des technologies supportées par les navigateurs web qui leur permettent d'être disponibles sur la plupart des plateformes. Cependant ces dernières reposent sur une gestion centralisée des données qui rend les utilisateurs très dépendants de la disponibilité de ces plateformes et d'une connexion internet pour la distribution des informations. Mises à part quelques exceptions, les fonctionnalités collaboratives sont souvent présentées comme mineures. Même si le « partage » de la visualisation à la manière des « réseaux sociaux » est assez courant, l'édition collaborative est souvent complexe à implémenter car les besoins sont nombreux. Parmi eux, on trouve tout d'abord le besoin d'avoir un système distribué conservant la cohérence des modifications de chacun des utilisateurs. Or le nombre d'utilisateurs ne doit pas affecter l'expérience ; le système doit supporter le passage à l'échelle. Un nombre d'utilisateurs élevé implique la mise en place d'un système de distribution de données adapté. Ce système est en plus contraint par la dynamicité des arrivées et départs des collaborateurs (*churn*) au cours d'une session. La dynamicité, qui ne permet pas d'avoir de pouvoir compter sur un nombre fixe de

ressources, doit s'accorder avec les besoins variables en termes de ressources. Chaque client est porteur de ressources rarement complètement exploitées lors d'épisodes collaboratifs.

La visualisation et la manipulation collaborative d'objets **3D** sont sujettes à plusieurs problématiques telles que la cohérence des ressources manipulées au cours du temps. Dans ce contexte, il existe un fort besoin de gérer l'évolution des versions permettant la revue des modèles **3D**. En s'appuyant sur les ressources à disposition, telles que les clients (navigateurs web) sur lesquels les utilisateurs manipulent les modèles **3D**, le passage à l'échelle est plus flexible car les ressources augmentent avec le nombre de clients et procurant également plus d'autonomie aux utilisateurs utilisant leurs ressources propres. L'architecture de communication et les contraintes liées aux architectures distribuées et décentralisées correspondent au second axe de recherche. Enfin, le dernier champ s'adresse à la partie que nous appellerons « métier » qui se rapporte au domaine de la manipulation d'objets **3D**, lequel inclut la gestion du cycle de vie des données **3D**, de l'interaction utilisateur au stockage en passant par leur distribution. Ces aspects se rapportent principalement à l'historique, la traçabilité de l'information et l'expertise embarquée dans le système.

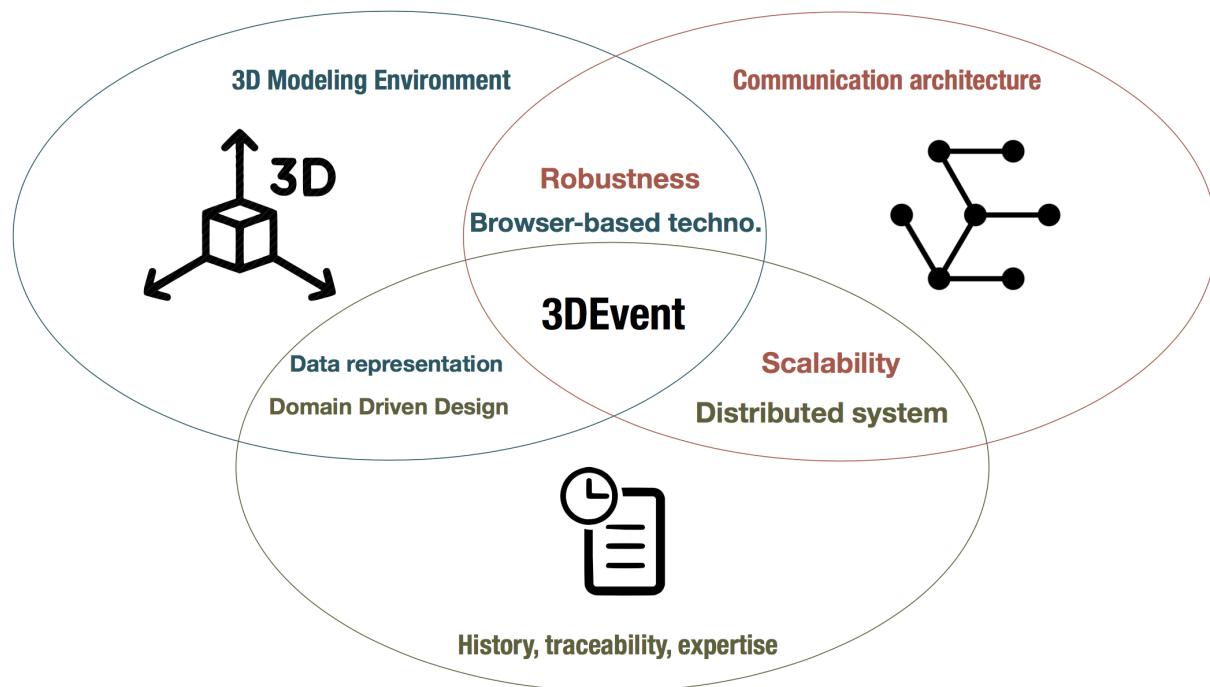


FIGURE 1.1 – Place de la contribution dans les différents champs de recherche

L'objectif de cette thèse est triple : (i) extraire les informations liées aux règles métier pour l'affichage et la manipulation d'objets **3D** en collaboration nécessaires à la traçabilité de l'information, (ii) repérer les principales problématiques de gestion des données sur le réseau pour avoir une transmission efficace et transparente pour l'utilisateur, (iii) proposer un **framework** pour un **EVC 3D** web intégrant ces contraintes réseau, métier, et **3D** dans

un navigateur. Dans le but d'atteindre ces objectifs, voici les cinq Questions de Recherche posées :

QR 1 Quelle architecture réseau est la plus adaptée pour une gestion efficace, robuste et temps réel des données **3D** dans un environnement web ?

QR 2 Quelle architecture logicielle confère une traçabilité des données conforme aux règles métier liées à la manipulation d'objet **3D** ?

QR 3 Quels sont les mécanismes assurant à l'utilisateur d'être autonome tout en ayant la possibilité de collaborer ?

QR 4 Comment faciliter l'implémentation d'un tel système en garantissant le respect des règles métier liés à la manipulation d'objet **3D** ?

QR 5 Quelles sont les métriques (réseau, collaboration) permettant d'évaluer un tel système de manière quantitative ? Qualitative ?

1.3 Contributions

La principale contribution de cette thèse est la définition et l'implémentation d'un ensemble de pratiques dans la gestion des données pour la manipulation d'objets 3D de manière collaborative sur web. Cela est très utile pour la gestion de l'historique d'une scène **3D** en utilisant des contenus **3D** de différentes natures. Dans ce but, j'ai choisi le paradigme événementiel et je l'ai intégré à mon modèle de collaboration grâce à l'utilisation de solutions préexistantes pour le partage et la diffusion de contenus **3D** dans le cadre d'applications collaboratives sur le web.

En constatant le manque de processus et d'outils qui pourraient réellement supporter la modélisation collaborative **3D** sur le web intégrant l'identification de problèmes spécifiques et leur formulation liés a également été importante. La recherche, cependant, introduit de nouveaux concepts dans le domaine de la gestion de données **3D**. Les contributions de cette thèse peuvent être résumées comme suit.

1.3.1 Contributions théoriques

- Proposition d'un système de gestion et de visualisation de contenu **3D** avec un historique non linéaire orienté événements.
- Définition d'une architecture hybride (client serveur et **P2P**) adapté à la distribution de contenus **3D** dans le cadre d'une collaboration sur web en temps réel.

- Introduction des concepts d'événements métiers liées à la **3D** comme moyen d'interagir dans une scène **3D** multi-échelle.

1.3.2 Contributions pratiques

- Définition d'une API ouverte et de l'application cliente utilisant les principes et les technologies du web.
- Implémentation d'un prototype 3DEvent Client et de son évaluation utilisateur.
- Implémentation d'un prototype 3DEvent Architecture et de son évaluation.

1.4 Organisation du manuscrit

La **3D** et les environnements virtuels collaboratifs **3D** sont de plus en plus présents dans l'industrie de la **CAO**, ce qui rend nécessaires de nouveaux modèles, outils et méthodes facilitant la mobilité et l'autonomie des utilisateurs. Le contenu multimédia **3D** est également devenu omniprésent dans notre quotidien. De ce fait, développer de nouveaux outils pour créer manipuler et partager du contenu **3D** est crucial. Le chapitre présente une architecture de communication hybride permettant de faciliter la transmission des objets **3D** en temps réel de manière transparente pour l'utilisateur. La création d'**EVC 3D** sur le web nécessite de nouvelles architectures tirant partie des nouveaux standards du **World Wide Web Consortium (W3C)**. Puis le chapitre présente deux contributions reposant sur l'architecture cliente de 3DEvent pour manipuler des objets **3D** avec une grande traçabilité et de manière autonome. Finalement, le chapitre aborde le problème spécifique de la transmission du contenu **3D** lors d'une session collaborative en temps réel. En considérant et développant un modèle orienté événements, 3DEvent dérive des stratégies pour délivrer et obtenir une scène **3D** en continu (*streaming* en anglais) en s'appuyant sur les différentes configurations matérielles, utilisateurs et réseaux.

Chapitre 2

État de l'art

Sommaire

2.1 Modélisation 3D collaborative sur le web	17
2.1.1 Collaboration et concurrence en 3D	17
2.1.2 ... sur le web : HTML5 et au-delà	19
2.1.3 Web 3D : deux approches	21
2.2 Communication en temps-réel	24
2.2.1 Fiable versus Non Fiable	26
2.2.2 Ordonné versus Désordonné	27
2.2.3 Les principaux protocoles de transport	28
2.2.4 Web et P2P : WebRTC	30
2.2.5 Quel protocole dans quel EVC3D ?	33
2.3 Les systèmes distribués orientés événements pour la collaboration	35
2.3.1 Introduction	35
2.3.2 Systèmes Publish-Subscribe	36
2.3.3 Domain Driven Design	39
2.3.4 Command Query Responsability Segregation	42
2.3.5 Event Sourcing	43
2.4 Conclusion	48

2.1 Modélisation 3D collaborative sur le web

2.1.1 Collaboration et concurrence en 3D

Dans un contexte collaboratif, on considère qu'il n'est pas possible pour une personne seule de réaliser la tâche proposée complètement. La modélisation 3D collaborative permet à différents concepteurs de travailler ensemble sur une même tâche de manière efficace (en terme de coûts temporel et financier) et avec un support visuel manipulable, éditabile et flexible (proposant différentes possibilités de visualisation). En se complétant, leur travail peut aussi entrer en conflit, se compenser, s'annuler... Mettre en place une situation de travail collaboratif demande en plus de prendre en compte de nombreux facteurs comme le type de collaboration (synchrone, asynchrone) et de cohérence souhaitées (forte, éventuelle) par exemple.

Mettre en place une situation de travail collaboratif demande en plus de prendre en compte, les différentes solutions pour la conception collaborative peuvent être divisées en trois types : (i) les mécanismes de type autoritaire (requérant des autorisations) ; (ii) les opérations de transformation ; (iii) et les autres solutions synchrones et asynchrones. Selon leur comportement face à l'apparition d'un conflit, ces types de collaboration sont classifiés comme pessimistes ou optimistes (Tableau 2.1). Les solutions de type autoritaire (« *authoritative* ») sont comptées parmi les solutions pessimistes car elles préviennent toute occurrence de conflit, tandis que les solutions des deux autres types, transformées opérationnelles – **Operational Transformation (OT)** – et autres (synchrones ou asynchrones), sont dites optimistes car permissives face à l'apparition de conflits. Dans ce dernier cas, les conflits détectés peuvent, selon les stratégies, amener à une résolution automatique ou manuelle.

La plupart du temps, ce sont des mécanismes autoritaires comme le mécanisme de verrouillage (*lock mechanism*), des feux de circulation (*traffic light mechanism*) requérant des droits ou des permissions temporaires, qui sont proposés dans les solutions collaboratives traditionnelles. Les utilisateurs doivent souscrire aux permissions du système avant de pouvoir effectuer leurs modifications. Ces mécanismes permettent de s'assurer que les opérations sont effectuées de manière séquentielle, ce qui assure la cohérence des données à travers le système. Plus simples à mettre en place, ces mécanismes sont aussi plus restrictifs vis-à-vis de la liberté de création des utilisateurs du système. Par exemple, Lets3D [Ha *et al.*, 2015] est un éditeur collaboratif 3D sur le web dont la gestion de la concurrence repose sur la demande de permission par objet sélectionné (ce qui verrouille l'objet pour les autres utilisateurs lorsqu'elle est accordée).

Tableau 2.1 – Solutions conventionnelles pour les problèmes collaboratifs [Yu et al., 2016].

Type	Mécanismes de Collaboration	Problème résolu
Autoritaire	Verrou Feux de circulation Droits d'accès Permission temporaire	Évitement des conflits
Transformées opérationnelles	Séquence de transformation Transformées opérationnelles 3D	Résolution de conflit temps réel pour les documents textuels Résolution de conflit pour des modèles 3D
Autre (synchrone ou asynchrone)	Semi-synchrone Signalisation des conflits Résolution de conflit créative	Détection de conflit ou résolution avec interaction utilisateur

Avec ceux d'Ellis et Gibbs, les travaux de Greenberg et Marwood sont parmi les premiers à s'intéresser aux problèmes liés à la gestion de la concurrence dans un système distribué au sein d'un collecticiel [Ellis et Gibbs, 1989, Greenberg et Marwood, 1994]. L'**EVC 3D Duplex** [Pacull et al., 1994], présenté au même moment, s'intéresse plus spécifiquement aux problématiques de passage à l'échelle en essayant de réduire la taille du contexte partagé pour limiter les problèmes liés à la concurrence. Les approches pour défaire / refaire (*undo/redo*) des manipulations dans les éditeurs collaboratifs distribués sont nombreuses. Les travaux les plus aboutis s'intéressent à la cohérence dans un environnement distribué collaboratif concernant les documents textuels [Prakash et Knister, 1994, Ressel et al., 1996, Sun, 2002]. Les algorithmes souvent utilisés dans ce cadre sont les algorithmes de transformées opérationnelles – **OT** – [Ellis et Gibbs, 1989] ou les algorithmes de réPLICATION de données commutatives – **Commutative Replicated Data Types (CRDTs)** – [Shapiro et Preguiça, 2007] s'orientant de plus en plus vers des solutions décentralisées pour un usage à grande échelle [Weiss et al., 2009].

L'existence de protocoles agnostiques¹ comme Dat [Ogden et al., 2017] permet également une synchronisation fiable pour des contenus dynamiques. Dat est un protocole de synchronisation de jeu de données qui n'assume pas que le jeu de données soit statique ou que le jeu de données soit entièrement téléchargé. C'est un protocole agnostique dont la conception repose sur la garantie de trois principes. D'une part, l'intégrité du contenu est garanti grâce à l'utilisation de valeurs de hachage (*hash*) signées. Elles sont mises à jour régulièrement de manière décentralisée (*decentralized mirroring*). Cela permet de

1. Indépendant des protocoles de communication, le protocole dit « agnostique » négocie le protocole avec son pair et commence la communication. Cette démarche apporte plus de flexibilité et d'interopérabilité au système.

découvrir automatiquement les pairs et effectuer l'échange de données en essaim. D'autre part, le protocole impose également un chiffrement des données de bout en bout. Enfin, il propose un versionnage incrémental pour la synchronisation des données. Ce protocole est, par exemple, utilisé par hashbase.io, un service qui se propose d'être un pair toujours présent pour l'hébergement de sites web en P2P, promettant ainsi une disponibilité pérenne du contenu.

2.1.2 ... sur le web : HTML5 et au-delà

L'arrivée du web a bouleversé les usages liés à la collaboration sur des objets 3D. Les principes, les technologies, l'omniprésence du web en ont fait une plateforme de prédilection pour la visualisation et la manipulation d'objet 3D de haut niveau. Pour les projets d'Architecture, Ingénierie et Construction (AIC) ou de BIM, la demande de traitement et la fidélité ont tendance à être plus élevés, particulièrement pour la représentation de dessins d'ingénierie ou de modèles d'architecture 3D. Les objets simples (primitives) ou les maillages optimisés pour le rendu temps réel ne sont ni suffisants ni assez génériques pour être supportés par tous les processus liés à l'élaboration d'un produit. Plus les projets sont gros, plus ils dépendent d'une multitude de logiciels 3D – dont l'interopérabilité n'est pas garantie – pourvoyant chacun aux besoins d'une tâche spécifique. Chaque tâche (ex : modélisation CAO, *stress testing...*) possède sa propre représentation des données qui peut varier selon les niveaux de sémantique attachés à la géométrie. Pour garder une trace de ces données et mettre en commun les données générées autour d'un produit, l'utilisation d'un Product Data Management (PDM)/PLM est souvent requise. Certains outils de création d'objets 3D (par exemple Autodesk Revit) autorisent – mais pas systématiquement – la synchronisation de fichier via leur dépôt à distance. En cela, une plateforme web pour un PDM/PLM a l'avantage d'être distribuée pour être accessible depuis un navigateur web ; avec un système d'édition hors-ligne, le téléchargement peut s'avérer long et fastidieux. De plus, les logiciels spécialisés pour la modélisation 3D, contrairement aux jeux multijoueurs, sont traditionnellement dédiés à un utilisateur unique sans représentation visuelle des autres opérateurs dans le même espace 3D. Ici encore, les principes d'accessibilité du web facilitent la création d'un espace virtuel 3D collaboratif pour la visualisation, la manipulation et l'échange de données partagées.

La collaboration en temps-réel est de plus en plus présente sur le web pour différents types de documents, notamment 3D. La revue sur la visualisation distribuée, effectuée par Grimstead et al. en 2005 [Grimstead et al., 2005], indiquait que la plupart des systèmes sont conçus pour moins de cent utilisateurs simultanés et reposent sur un ou plusieurs

serveurs pour supporter ces utilisateurs. Ils expliquent ce schéma par la volonté du fournisseur de service d'assurer une qualité de service et la sécurité du système.

Les systèmes de **Réalité Virtuelle (RV)** collaboratifs font exception à ce schéma où l'utilisation des réseaux **P2P** est plus répandue pour supporter parfois plus d'un millier d'utilisateurs simultanés. Dans chacun des cas, chaque système a besoin d'un client adapté pour opérer de manière isolée, sans inter-opérer avec les autres systèmes. C'est dans ce contexte qu'en 2011, Mouton et al. [Mouton et al., 2011] présentent une analyse approfondie de l'état de l'art des environnements collaboratifs **3D**, ciblant principalement la visualisation collaborative. Ils montrent l'apparition de la tendance à déporter les environnements collaboratifs sur le web, grâce à l'évolution d'XMLHttpRequest en client-serveur et l'apparition du standard HTML5 comprenant un support avancé de l'audio et de la vidéo, ainsi que plusieurs **Application Programming Interface (API)** de stockage côté client (LocalStorage, IndexedDB).

De plus, les applications web revêtent plusieurs avantages par rapport aux applications natives sur mobiles ou aux logiciels autonomes. Cela repose principalement sur le fait que les navigateurs sont présents partout dans nos vies en 2017, incluant les téléphones intelligents et les tablettes qui les rendent indépendants des plateformes utilisées. Le déploiement sur le web ne requiert pas d'installation ou de mises à jour autres que celles du navigateur – en ne considérant que les applications sans greffon (*plugin less*). La modification de l'application est gérée de manière centralisée par les serveurs qui distribuent l'application. Les éditeurs peuvent diffuser leur application à l'échelle mondiale instantanément et la mettre à disposition des utilisateurs sans dépendre d'un réseau de distribution autre qu'internet.

Parmi les solutions sans greffon dédiée à de la modélisation **3D**, de nombreuses utilisent l'architecture client-serveur et le protocole **WebSocket** pour effectuer la synchronisation entre les différents clients collaborateurs. Quelques raisons peuvent expliquer cette situation. Historiquement, les développeurs sont plus familiers avec les architectures client-serveur et le standard **WebSocket**. Techniquement, la gestion de données (la synchronisation et la gestion de la concurrence) est facilitée dans un **Environnement Virtuel Collaboratif (EVC)** avec une architecture centralisée. Commercialement, le suivi des données de l'utilisateur est plus simple avec un serveur central qui garantit un engagement de l'utilisateur vis-à-vis du service (voire une dépendance).

Le choix de plateformes web dans l'infonuagique (« cloud ») pour la modélisation **CAO** est prépondérant pour les solutions commerciales comme OnShape, Clara.io et Tinker-CAD. Les fonctionnalités minimum de ces plateformes consistent en la mise en place d'un environnement **3D** pour visualiser un objet **3D** accompagnées d'un système défaire / refaire

et d'une plateforme de diffusion des créations. OnShape est un service de modélisation 3D très riche dont l'objectif est de proposer une qualité équivalente des fonctionnalités de modeleurs autonomes (*standalone*) professionnels, le tout de manière collaborative. Leur approche de la gestion de version développe le concept de micro version [Baran, 2015]. Cette approche est employée dans les travaux de Lu et al. [Lu et al., 2016] pour gérer l'évolution de productions participatives liées à des tâches de modélisation 3D en usant de la créativité, de l'intelligence et du savoir-faire d'un grand nombre de personnes en sous-traitance (*crowdsourcing*). Le modeleur 3D Clara.io [Houston et al., 2013] est plus généraliste. Il s'oriente davantage vers des fonctionnalités avancées de rendu 3D sur le *cloud* en utilisant du lancer de rayons. Les fonctionnalités d'historisation et de collaboration proposées dans Clara.io restent très basiques (seulement annuler / refaire). TinkerCAD est aussi une plateforme de publication d'objets 3D pour faciliter l'accès à la conception d'objets 3D. TinkerCAD est une application destinée au grand public pour la conception et l'impression 3D. De ce fait, les fonctionnalités sont limitées à cause du domaine (l'impression 3D) et de la cible (le grand public). La gestion de version est donc plus légère, similaire à celle proposée par Clara.io. OpenJSCAD, Verold Studio, Vectary sont d'autres exemples de modeleurs 3D moins connus, avec des fonctionnalités proches de TinkerCAD. Pour réduire l'empreinte mémoire des messages transmis lors de la collaboration sur le web, certains systèmes d'édition collaborative de contenu 3D utilisent une modélisation par surfaces implicites (BlobTree) [Grasberger et al., 2013]. Les objets sont alors manipulés comme des géométries de construction de solides avec les opérations booléennes associées. Le tableau 2.2 résume les différents types d'encodages associés aux types de modélisation 3D.

Tableau 2.2 – Encodage des données transmises

Référence	Modélisation 3D	Encodage des modifications
[Grasberger et al., 2013]	Blob (CSG)	Fonction implicite
Clara.io [Houston et al., 2013]	Polygonale	Commande interface
[Mouton et al., 2014]	Polygonale	Fonction paramétrique
OnShape [Baran, 2015]	Paramétrique	Microversion
cSculpt [Calabrese et al., 2016]	Polygonale	Fréquence spatiale

2.1.3 Web 3D : deux approches

La plateforme web possède certains avantages par rapport à des clients lourds. En effet, l'essor de l'infonuagique a permis le développement de meilleures infrastructures de

services. Ces dernières ont rendu faisable, en termes de ressources, mais également en termes de technologies la création de plateformes collaboratives sur le web pour faire de la conception 3D.

Il existe deux types d'approches pour créer du contenu web 2D ou 3D : l'approche déclarative et l'approche impérative. La Figure 2.1 montre le pendant 3D pour chaque approche 2D. En 2D, les spécifications déclaratives ([Scalable Vector Graphics \(SVG\)](#)) et impératives ([canvas](#)) sont issues du dernier standard HTML (HTML5), alors qu'en 3D, les spécifications sont encore en évolution.

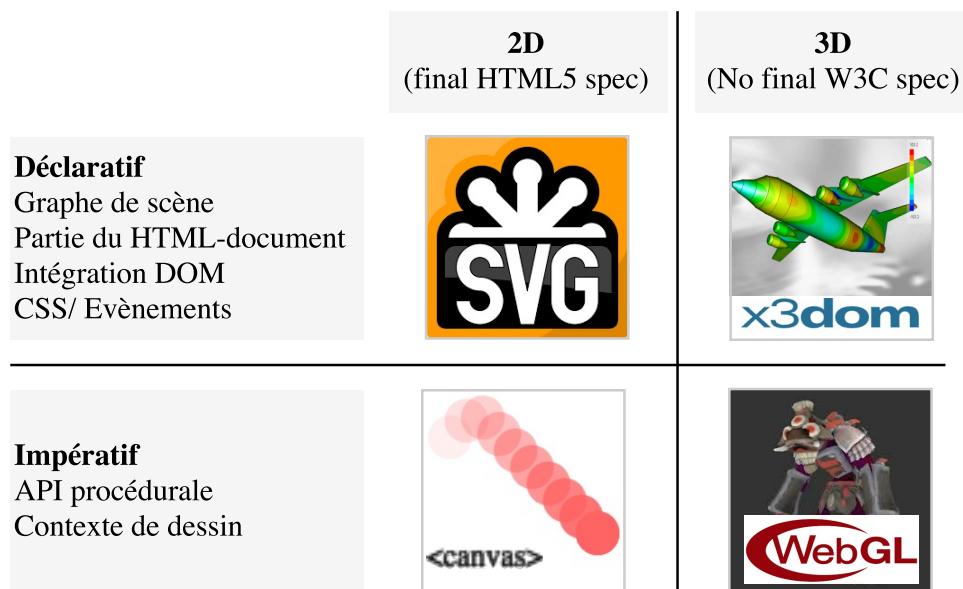


FIGURE 2.1 – Déclaratif vs Impératif en 2D et en 3D sur le web

L'approche déclarative (Dec3D) s'intègre au [Document Object Model \(DOM\)](#) et se focalise sur l'utilisation de technologies du web existantes comme CSS3, HTML5 et l'Ajax. X3D est ainsi un format de fichier respectant le standard ISO [W3C, 2011] permettant de représenter des scènes 3D interactives en XML et rétro-compatibles avec VRML97. Il se différencie des formats de fichiers comme Collada en intégrant le comportement de la scène durant l'exécution en plus de la description du contenu 3D. Les deux bibliothèques les plus notables dérivées de ce standard sont X3DOM [Behr *et al.*, 2010] et XML3D [Sons *et al.*, 2010] : toutes les deux sont capables de supporter les récentes avancées de WebGL pour afficher une scène décrite dans le [DOM](#) à l'intérieur d'un *canvas* HTML5. X3DOM essaye de respecter le standard X3D et ses concepts pour que le format s'intègre au [DOM](#). De plus, X3DOM intègre le support d'éléments [HyperText Markup Langage \(HTML\)](#), les événements [DOM](#) et de profils [CSS](#) en supplément [Sutter, 2015]. En comparaison avec X3DOM, XML3D a développé une extension à [HTML5](#) pour décrire une scène 3D. L'utilisation d'un langage déclaratif comme X3D s'adapte bien à des contextes

où le XML est très présent (exemple : Système d'Information Géographique (SIG)) : le contenu 3D peut être directement transformé (ex. XSLT) d'une représentation à une autre (tout comme le X3D peut être rendu dans un navigateur en utilisant X3DOM).

L'approche déclarative est utilisée dans de nombreux travaux de visualisation scientifique 3D distribuée [Jung *et al.*, 2012], par exemple, pour des données spatiales [Stein *et al.*, 2014] ou du rendu volumique [Becher, 2012]. En manipulant directement les objets 3D à partir des éléments DOM, grâce à sa structure bien connue, il est alors plus simple, lors de la collaboration, d'utiliser ces éléments [Gadea *et al.*, 2016] ou les événements du DOM [Lowet et Goergen, 2009] comme protocole de synchronisation de scènes. L'ajout de nombreux *listeners* sur un document peut affecter les performances de parcours de l'arbre du DOM notamment si une scène est très peuplée.



FIGURE 2.2 – Support de WebGL 1 (2014-2017) et WebGL 2 (2016-2017)

Concernant l'approche impérative, la spécification WebGL 1.0 [Khronos, 2011] proposée par le groupe Khronos permet aux navigateurs d'effectuer un rendu 3D grâce à une API JavaScript adaptée de l'API d'OpenGL ES 2.0 [Khronos, 2007] en utilisant l'élément *canvas* d'HTML5. Cette spécification est supportée par la plupart des navigateurs traditionnels (Chrome depuis v49, Firefox depuis v52, Safari depuis v9.1, IE 11², Edge 14² et Opera depuis v41) ainsi que les navigateurs mobiles (iOS Safari depuis v9.3, Android Browser depuis v53 et Chrome for Android depuis v53) à l'exception d'Opera Mini³. La spécification WebGL 2.0 [Khronos, 2016] basée sur OpenGL ES 3.0 [Khronos, 2008], déjà publiée en tant que brouillon, est en phase expérimentale⁴. La Figure 2.2 montre l'évolu-

2. Le contexte WebGL est accessible depuis « `experimental-webgl` » au lieu de « `webgl` ».

3. Données issues de <http://caniuse.com/#search=webgl>, consulté le 26/11/2016.

4. <http://caniuse.com/#feat=webgl2> consulté le 26/11/2016

tion du support de WebGL 1 et WebGL 2 durant ces dernières années sur les différentes plateformes possédant un navigateur web⁵.

2.2 Communication en temps-réel

L'expression temps-réel est largement utilisée par des applications requérant une forme de temps de réponse rapide et réactif pour que l'utilisateur ait une bonne expérience. La communication dans un EVC est primordiale pour échanger entre les collaborateurs. Elle doit respecter l'intention des utilisateurs pour leur permettre de s'immerger dans la collaboration et faire confiance à l'application (exactitude des modifications).

L'exigence vis-à-vis des temps de réponse est grande et ce pour deux raisons principales : les limitations humaines (mémoire et attention limitées à court terme) et les aspirations de l'être humain (besoin d'être en contrôle sur les machines). S'accroissant au gré de la technologie et des attentes des utilisateurs (rétro-compatibilité, temps de chargement), cette exigence varie selon le domaine : elle est prépondérante sur le web en général. Par exemple, dans un domaine connexe comme le e-commerce, une étude réalisée en 2009 explique qu'une grande partie des internautes abandonneraient leurs achats en ligne si les pages mettaient 2 secondes ou plus à charger⁶. De nos jours, ce délai a été réduit au quart de seconde pour les grandes entreprises du web. Jakob Nielsen [Nielsen, 1993] indique quatre valeurs limites concernant les temps de réponses :

- *0,1 seconde* donne une sensation de réponse instantanée, comme si le résultat avait été produit par l'utilisateur et non l'ordinateur. Ce niveau de temps de réponse soutient la sensation de manipulation directe.⁷
- *1 seconde* garde le flux de pensées de l'utilisateur sans interruption. L'utilisateur peut ressentir un délai et par conséquent savoir que c'est la machine qui génère le résultat ; il a quand même une impression de contrôle sur l'expérience générale et peut se déplacer librement dans l'interface sans attendre la machine. Ce degré de réactivité est impératif pour une bonne navigation.

5. Images capturées sur <https://webglstats.com/>. Les statistiques sont collectées à partir de sites partenaires de webglstats.com. Ces sites ciblent des néophytes de WebGL possédant en général le matériel dédié à la 3D. Consulté le 04/07/2017.

6. <https://www.akamai.com/us/en/about/news/press/2009-press/akamai-reveals-2-seconds-as-the-new-threshold-of-acceptability-for-ecommerce-web-page-response-time.jsp>

7. En IHM, la manipulation directe correspond à un mode d'interaction au cours duquel les utilisateurs font des actions sur les objets d'intérêt affichés dans l'interface utilisateur en utilisant des actions physiques, incrémentales et réversibles dont les effets sont immédiatement visibles sur l'écran.

- 10 secondes conserve l'attention de l'utilisateur. Entre 1 et 10 secondes, l'utilisateur se sent dépendant de la machine, mais peut faire avec.
- Au delà de 10 secondes, l'utilisateur va commencer à penser à autre chose, rendant difficile le retour à la tâche une fois que la machine répond.

Dans le domaine de l'édition et la manipulation collaborative d'objets 3D, les contraintes abordées se situent à divers degrés : au chargement et lors des mises à jour. Le chargement concerne la phase de téléchargement des ressources (page web, modèles 3D), parfois lourdes. Cette phase peut tolérer des délais relativement longs (1-10s) compte tenu du fait que l'utilisateur connaît en partie les contraintes liées à la taille des objets 3D. Concernant les mises à jour, l'édition collaborative requiert un temps de réponse raisonnablement court qui est contraint par l'exactitude des calculs et le temps dans lequel le résultat est produit. Pour les mises à jour internes – produites par l'utilisateur, actions sur l'interface – on s'accordera dans cette thèse sur un délai inférieur à 0,1 seconde. Pour les mises à jour externes – produites par les collaborateurs – les latences réseaux rentrent en compte (serveur, pairs...), et on s'accordera sur un délai entre 1 et 10 secondes selon les degrés d'asynchronicité possibles.

La communication et la transmission des données lors de la collaboration comptent parmi les piliers d'un EVC 3D. Que ce soit dans un Environnement Virtuel (EV), un jeu multijoueur ou un jeu sérieux (*serious game*), les participants interagissent en (quasi) temps réel, alors qu'ils sont situés à différents endroits géographiques. Le développement d'un EVC 3D nécessite donc des connaissances issues de plusieurs domaines comme la conception et l'implémentation de protocoles réseaux et de réseaux distribués, la gestion des données concurrentes, la visualisation 3D et l'ergonomie.

Dans le monde du réseau, un protocole de communication en temps-réel existe, il s'appelle Real-Time Transport Protocol (RTP) et a été développé dans le but de faciliter la communication en temps-réel sur un réseau. Coexistant avec RTP, le protocole Real-time Transfert Control Protocol (RTCP) est utilisé pour transmettre des informations de contrôle à propos des données en temps-réel. RTCP transporte des données contenant des informations de contrôle et des informations statistiques concernant les flux de données et les connexions des destinataires qui permettent à l'expéditeur d'ajuster le flux en conséquence. Les protocoles RTP et RTCP sont principalement conçus pour la transmission de flux audio et vidéo, moins pour des données arbitraires comme des données 3D. Pour cette raison, leur utilisation est limitée dans le cadre des EVC pour la modélisation 3D.

La variété des domaines et applications auxquels s'adressent les EVCs 3D actuels fait qu'il n'existe pas de protocole de communication parfait, adapté à tous les types

d'environnements. Les besoins en communication, qui se formulent souvent en termes de fiabilité de la livraison des messages et de l'ordonnancement des messages, sont les variables d'ajustement en fonction des contraintes qu'imposent les limites de temps de réponse et de bande passante.

2.2.1 Fiable versus Non Fiable

La fiabilité d'un protocole de communication définit comment le protocole fait face à la perte ou au retard de données. Un protocole dit « fiable » (*reliable*) fait en sorte de rendre sûr le fait que le destinataire recevra éventuellement les données envoyées par l'expéditeur. Lors de la perte d'un paquet, ou lorsqu'il est retenu quelque part dans le réseau, le protocole est au courant et essaye de renvoyer les données concernées. Un protocole dit « non fiable », ne fournit pas ce genre de garantie et n'essaiera pas de transmettre à nouveau les données.

Bien que la fiabilité dans une communication soit une propriété habituellement recherchée, elle implique également que le canal doive connaître l'état des données qui transitent. Dans le cas où l'expéditeur numérote les paquets en séquence, seul le destinataire a la connaissance des paquets manquants (et à renvoyer). Pour indiquer les paquets manquants à l'expéditeur, le destinataire peut envoyer des acquittements (ACKs) pour confirmer leur réception. Ces ACKs peuvent être joints aux messages ou envoyés séparément. En plus des données additionnelles envoyées, un protocole fiable peut aussi générer des latences sévères (dans l'attente d'un paquet perdu ou avec beaucoup de retard). Cela peut impliquer que les « nouvelles » données peuvent arriver et être déjà périmées.

Pour certaines applications qui ne peuvent pas faire face à ces délais potentiels ou ne peuvent pas se permettre l'utilisation de données additionnelles, les protocoles non fiables sont utilisés. L'expéditeur se repose alors sur le réseau pour délivrer les paquets correctement, mais il n'y a aucune garantie. Les protocoles non fiables ne nécessitent pas de données additionnelles (séquence de nombre ou ACKs). L'expéditeur n'est cependant jamais sûr que le destinataire a reçu toutes les données envoyées. Lorsque le destinataire a une connaissance du type de trafic qu'il reçoit, il peut fournir un retour à l'expéditeur pour qu'il ajuste le flux de paquets. Ce principe est appliqué par le protocole [RTCP](#), conjointement avec [RTP](#), pour fournir les données statistiques liés à la connexion à l'expéditeur.

L'utilisation d'un protocole de communication fiable ou non fiable dépend du type d'application et du type de l'environnement réseau dans lequel il est conçu pour opérer. Si une application dite temps-réel est conçue pour fonctionner en réseau local – [Local](#)

Area Network (LAN) – l'utilisation d'un protocole fiable est une option recommandée. Les câbles utilisés dans ce contexte sont souvent rapides et stables ; un paquet perdu ou corrompu peut rapidement être détecté et retransmis. Dans des environnements réseaux moins prévisibles comme les **Metropolitan Area Network (MAN)** ou **Wide Area Network (WAN)**, la nature de l'application a une influence importante sur le type de protocole à utiliser. Selon si la pertinence des données est de courte durée, ou si la perte de données n'est pas aussi importante que le fait de la retarder, alors un protocole non fiable peut être employé.

2.2.2 Ordonné versus Désordonné

Les paquets envoyés par l'expéditeur ne prennent pas forcément tous le même chemin vers le destinataire. La réaction des réseaux face à la congestion consiste à modifier les tables de routage afin que les paquets évitent le noeud congestionné. Cela peut altérer l'ordre des paquets à la réception. Le choix du protocole a également une influence sur la gestion de l'ordre d'arrivée des paquets.

Les protocoles dits « ordonnés » garantissent que l'ordre dans lequel les paquets sont envoyés est préservé dans l'application lors de la réception de ces données par le destinataire. Même si les paquets arrivent dans le désordre, le protocole peut imposer l'ordonnancement des données à l'aide d'une mémoire tampon en attendant les données précédentes. Cela impose que le protocole (comme pour un protocole fiable) numérote les paquets ou utilise un estampillage pour connaître l'ordre. Cependant, la fiabilité n'est pas nécessaire pour la livraison dans l'ordre. Par exemple, dans une conversation vidéo, si une image met trop de temps à arriver, elle sera mise en tampon, puis omise au-delà d'une limite (taille tampon ou temps).

Forcer à la fois la fiabilité et l'ordonnancement peut causer des latences du côté du destinataire, même lorsqu'une grande partie des données est déjà reçue. Quand un des premiers paquets est perdu ou en retard mais que d'autres données ont déjà été reçues, ces données doivent attendre avant d'être livrées, tant que le paquet n'est pas arrivé. Les communications non ordonnées sont plus simples à gérer car les données sont délivrées dès qu'elles arrivent. Cela signifie également que la numérotation des paquets n'est pas requise (paquets plus légers). L'application est alors en charge de gérer le fait que les paquets arrivent dans le désordre. La livraison de paquets ordonnés est souvent une fonctionnalité appréciée dans le cadre des **Système d'édition Collaborative (SEC)**. C'est particulièrement le cas en **3D** afin de respecter l'intention de l'utilisateur, car l'ordre des

actions en détermine le résultat. Par exemple, le calcul deux rotations consécutives se traduit par la multiplication deux matrices **3D** qui n'est pas commutatif.

2.2.3 Les principaux protocoles de transport

Les principes de fiabilité et d'ordonnancement décrits plus haut sont souvent utilisés pour discriminer les différents protocoles de communication. En ce qui concerne les applications en temps-réel, ces principes sont considérés comme les aspects les plus importants à considérer pour le choix d'un protocole de communication. La présentation succincte des protocoles ci-dessous indique leur propriété de fiabilité et d'ordonnancement.

TCP

Le **Transmission Control Protocol (TCP)** est probablement le protocole le plus utilisé sur internet pour transporter des données : navigation web, chat, transfert de fichiers... **TCP** est un protocole dit « orienté connexion », ce qui implique que les deux terminaux sur le réseau s'informent et s'accordent chacun sur ce qu'ils veulent envoyer / recevoir de la part de l'autre. Quand cet accord est établi, la connexion **TCP** est ouverte et le flux peut transiter par la connexion. **TCP** envoie des données sous forme de flux continu d'octets qui arrivent de manière fiable et ordonnée. L'application peut lire les données à partir de ce flux.

TCP a également un champ de somme de contrôle (*checksum*) afin de vérifier les données dans les cas d'erreurs (mauvais signal, routeur défectueux). Si la somme est incorrecte, la données est écartée et considérée comme perdue. Une nouvelle copie de la donnée est alors attendue à la réception lorsque l'expéditeur détecte qu'il n'a pas reçu le ACK concernant la donnée erronée.

UDP

Le **User Datagram Protocol (UDP)** est considéré comme l'opposé de **TCP** concernant plusieurs aspects. Premièrement, **UDP** ne nécessite pas, pour établir la connexion, d'étape de configuration et d'accord comme **TCP**. Les deux terminaux doivent configurer un *socket* **UDP** et écouter ce *socket* pour récupérer les données entrantes. Du fait qu'aucun accord n'est établit par les deux terminaux, chacun peut envoyer ou recevoir des données de n'importe quel *socket* proprement configuré. Deuxièmement, **UDP** est un protocole dit « orienté message ». Par rapport à **TCP** où il n'y a pas de distinction claire entre les messages du fait de sa nature (flux continu), **UDP** fait une distinction claire entre chaque message (un message envoyé donne un message reçu, sauf si perte du message).

Les messages qui transitent en **UDP** sont envoyés de manière non fiable et désordonnée. L'en-tête **UDP** n'a pas de notion de numérotation, il est plus léger que **TCP** à cet égard.

UDP n'a pas de champ pour faire une somme de contrôle pour détecter les potentielles erreurs dans un paquet. Cependant, cette fonctionnalité est optionnelle dans IPv4 qui rapport une somme de contrôle remplie de zéros en cas d'erreur. En IPv6, la somme de contrôle est obligatoire ; donc même si **UDP** est non fiable, il doit fournir les moyens de vérifier les paquets pour la fiabilité de la transmission.

Même si **UDP** n'est ni fiable et ni ordonné, il est souvent utilisé comme protocole de base sur lequel d'autres protocoles viennent se greffer car il reste très léger. Ces autres protocoles peuvent s'assurer de l'ordonnancement et / ou de la fiabilité à leur manière (re-ordonnancement, retransmission...).

SCTP

Le **Stream Control Transmission Protocol (SCTP)** est le plus jeune des trois protocoles présentés. Ce protocole emprunte des fonctionnalités à **TCP** mais utilise une approche orientée message comme **UDP**. Comme **TCP**, la connexion doit être négociée entre les deux terminaux. **SCTP** discute également le nombre de flux utilisés : il autorise le multiplexage de plusieurs flux de données dans une seule connexion **SCTP**. En comparaison, **TCP** sépare chaque flux dans une connexion différente : le support de plusieurs flux est géré à plus haut niveau.

SCTP a été conçu comme un protocole fiable avec une option pour envoyer les messages de manière ordonnée ou non ordonnée. L'ordonnancement choisi est signifié par un *bit-flag* dans l'en-tête associée au message. Une extension du protocole permet de configurer la fiabilité d'un message expédié. La fiabilité partielle peut être établie en fonction de différents paramètres (temps d'attente ou nombre de tentatives avant retransmission). Cependant, seul l'expéditeur est au courant du niveau de fiabilité des données. Lorsque le destinataire remarque des données manquantes, il envoie à l'expéditeur des acquittements sélectifs (SACKs). Un SACKs contient les séquences de messages reçus et manquants et c'est à l'expéditeur de vérifier la configuration de la fiabilité des messages. Pour les messages marqués non fiables, l'expéditeur génère un nouveau paquet pour notifier le destinataire qu'il peut les « oublier » et avancer sa séquence de nombre à la valeur données dans le nouveau paquet.

Le champ de vérification de somme dans l'en-tête **SCTP** est deux fois plus grand que pour **UDP** ou **TCP** : 32 bits. Cela est dû au support des fonctionnalités comme le multiplexage qui introduit de nombreuses sous entêtes par exemple.

L'adoption de ce protocole est ralentie par le fait que l'implantation de **SCTP** n'est pas supportée par tous les systèmes d'exploitation notamment Windows [Hogg, 2012].

Tableau 2.3 – Aperçu des protocoles de transport

	Fiabilité	Ordonnancement	Error-free
TCP	Oui	Oui	Oui
UDP	Non / À plus ht niveau	Non / À plus ht niveau	Opt. (IPv4) / Oui (IPv6)
SCTP	Configurable	Configurable	Oui

2.2.4 Web et P2P : WebRTC

Le développement de la communication en **P2P** sur le web a débuté en 2011 avec les premiers brouillons du standard **Web Real-Time Communication (WebRTC)** proposé par le **W3C** et l'**Internet Engineering Task Force (IETF)**⁸. **WebRTC** est une technologie qui fournit aux navigateurs web la possibilité de communiquer en temps-réel (*Real-Time Communications*) via une collection de standards, protocoles et **APIs** JavaScript (Figure 2.3). L'un des atouts de cette technologie est de permettre de façon simple et sans module d'extension la capture d'un flux audio et / ou vidéo (ex : applications de VoIP), ainsi que l'échange de données arbitraires entre navigateurs sans nécessiter d'intermédiaires (ex : partage de fichier en P2P).

Techniquement, **WebRTC** supporte un canal temps-réel bidirectionnel pour l'échange de données. Contrairement à **WebSocket**, qui est basé sur **TCP**, **WebRTC** se base sur **UDP** en intégrant une pile de plusieurs protocoles (Figure 2.3) qui lui offre des fonctionnalités similaires (fiabilité, ordonnancement, sécurité).

Plusieurs projets impliquant les protocoles proposés par **WebRTC** se sont intéressés à l'**API MediaStream** (flux continus audio et vidéo) mais très peu utilisent la partie dédiée au transfert de données *DataChannel*. Selon le site officiel de WebRTC⁸, plus d'un milliard d'utilisateurs ont utilisé la technologie *open source* WebRTC. Ericsson Labs a été parmi les premiers à implémenter une application WebRTC en 2011. La jeunesse du protocole (2011) fait que peu de travaux académiques l'utilisent pour créer des **EVC 3D** [Desprat *et al.*, 2015b, Steiakaki *et al.*, 2016] et optimiser le partage de modèles 3D [Koskela *et al.*, 2014]. Côté commercial, un grand nombre d'applications et de services basés sur ce protocole ont fait leur apparition (par ordre d'importance) : des systèmes d'échanges audio / vidéo (VoIP)⁹, des systèmes de partage de fichiers¹⁰ et autres applica-

8. <https://webrtc.org>

9. WebRTCWorld en liste un peu plus de 140 <http://www.webrtcworld.com/webrtc-list.aspx>. Consulté le 07/07/2017.

10. WebTorrent <https://github.com/feross/webtorrent>. Consulté le 04/09/2017.

tions (capture d'écran, réalité augmentée, jeux)¹¹. WebRTC se retrouve également dans le domaine de l'optimisation de l'utilisation des ressources pour la création de Content Delivery Networks(CDNs) [Zhang et al., 2013], de grilles de calcul comme dans browserCloud.js [Dias, 2015] ou pour faire des analyses visuelles collaboratives dans un environnement hétérogène [Li et al., 2015].

L'utilisation et la gestion de réseaux P2P sont des changements de paradigme qui peuvent représenter une barrière d'entrée auprès des développeurs web, habitués aux architectures client-serveur. Cependant, l'implantation de WebRTC, très semblable à WebSocket, rend la technologie facilement abordable dans un premier temps. L'attrait pour les propriétés du P2P (distribution, résilience) est une motivation supplémentaire, en plus des implications sociales que cette architecture (partage, coopération, coût) dressent en faveur de l'intégration de WebRTC dans les applications web.

Les machines clientes et les terminaux d'extrême (end systems) sont considérés comme « les ressources » dans un réseau P2P. En principe, ces ressources sont difficilement administrables à l'échelle dans un réseau non structuré ; la qualité de service peut en pâtir car la synchronisation est plus complexe. C'est également problématique pour simuler ces environnements car il n'existe pas de solution pérenne pour tester WebRTC à l'échelle ; chaque service doit créer son propre système de test. Dans un contexte industriel, il est peu probable d'être confronté à une architecture P2P totalement décentralisée (sans serveur). Une solution centralisée permet d'utiliser la partie serveur pour télécharger l'application cliente qui contient la couche intergicielle P2P. Les serveurs sont aussi présents pour fournir à l'utilisateur différents services (ex : base de données, service de signalisation).

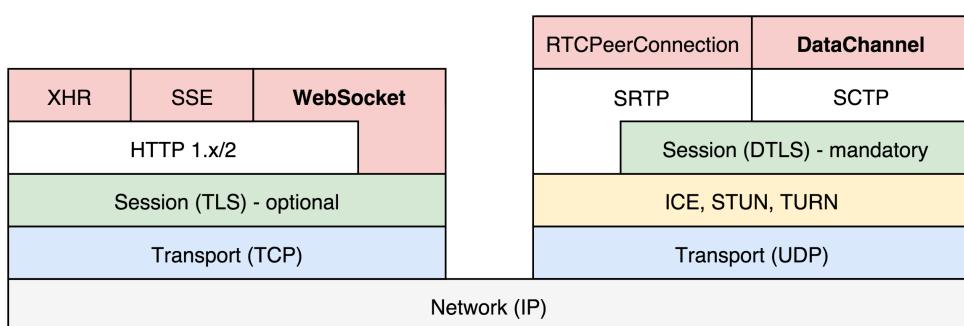


FIGURE 2.3 – Pile des protocoles IP : TCP vs UDP

11. Curation de ressources et modules WebRTC recensant plus de 100 projets <https://github.com/openrtc-io/awesome-webrtc>. Consulté le 07/07/2017.

Échange de données via RTCDataChannel L’[API RTCDataChannel](#) ou DataChannel, issue de WebRTC, permet l’échange P2P de données arbitraires avec peu de latence et un bon débit sur la base des sessions RTCPeerConnection. DataChannel utilise de multiples canaux simultanés avec une gestion de priorité. L’[API](#) est aussi capable de gérer la fiabilité de la livraison des paquets et sécurise nativement les données avec le protocole [Datagram Transport Layer Security \(DTLS\)](#). Le contrôle de la congestion des flux et l’ordonnancement éventuel est effectué par les protocoles [SCTP](#) et [Receiver-side Real-Time Congestion Control \(RRTCC\)](#).

Depuis 2011, les contributeurs du standard WebRTC font beaucoup d’efforts pour élargir et maintenir la compatibilité et interopérabilité entre les navigateurs. Cependant, certains navigateurs (comme Chrome) imposent une limite de taille pour l’envoi de messages contenant de la donnée brute, ce qui contrevient au principe énoncé par le standard.

Mécanisme de signalisation et problématiques réseaux

Le mécanisme de signalisation (*signaling mechanism* en anglais) permet de coordonner et d’envoyer des messages de contrôle durant une session. Le mécanisme de signalisation est utilisé pour échanger trois types d’information :

- des messages de contrôle sur la session pour ouvrir ou fermer un lien de communication et rapporter les erreurs ;
- les configurations réseaux (ICE candidates, IP ou port de l’ordinateur) ;
- et les capacités média du navigateur (limite de débit, codecs, résolution, formats de donnée).

Un service de signalisation est nécessaire à la mise en relation des différents pairs au cours d’une session collaborative. Pour établir un canal entre deux navigateurs (initialiser une session WebRTC), il faut que chaque client se « signale » l’un à l’autre. Techniquement, la connexion est établie en passant par un tiers ou directement entre deux pairs une fois qu’ils ont reçu leurs identifiants de connexion. Une fois que la connexion est établie entre deux pairs, RTCDataChannel est capable, en théorie, d’être le canal de signalisation. Cette solution peut aider à réduire la latence (car les messages sont envoyé directement entre les pairs) mais n’est pas très fiable. L’utilisation d’un tiers (serveur) est, en pratique, quasi systématiquement privilégiée.

La Figure 2.4 représente le cas le plus complexe que l’on peut rencontrer pour établir une connexion WebRTC. Le plus simple des réseaux P2P peut être résumé comme deux machines connectées directement l’une à l’autre. Cette vision un peu utopique ne tient pas compte des routeurs qu’une connexion internet peut rencontrer. Chaque routeur peut

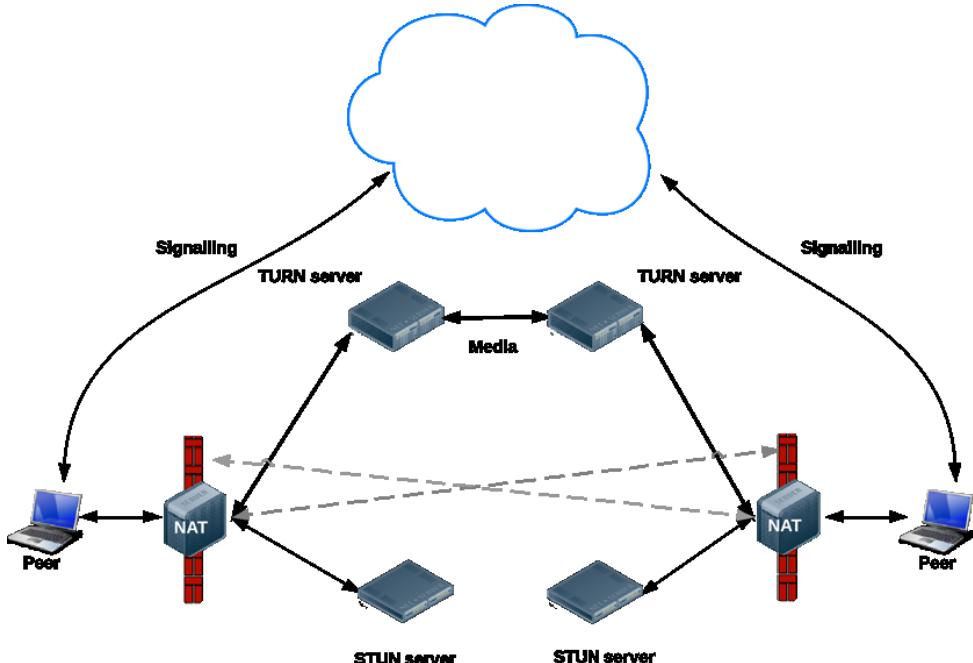


FIGURE 2.4 – STUN, TURN et signalisation

posséder un **Network Address Translator (NAT)** afin de faire correspondre les adresse IP à d'autres adresse IP (dans le cas d'un intranet par exemple). Cela permet de faire communiquer des machines non uniques et non routables en faisant semblant d'utiliser des adresses externes, uniques et routables. Pour permettre à un client situé derrière un routeur **NAT** de connaître son adresse IP publique et le type de serveur **NAT**, l'utilisation d'un serveur **Simple Traversal of UDP through NATs (STUN)** est requise. L'utilisation d'une **NAT** dynamique impose des restriction sur l'adresse de destination. Dans ce cas, le **STUN** est inutile, on utilise un serveur **Traversal Using Relays around NAT (TURN)** qui est considéré comme une extension du **STUN** et fait office de proxy. Une fois le client parvenu à récupérer une adresse IP, il peut participer à l'initialisation d'une connexion WebRTC (signalisation). L'utilisation de protocoles tels que **Interactive Connectivity Establishment (ICE)** avec son **ICE Framework** permet de faciliter la mise en relation des clients derrière un **NAT**. **ICE Framework** choisi le chemin le plus cours entre les deux pairs et va utiliser le protocole adapté pour établir la connexion (**STUN** d'abord puis en cas d'échec, **TURN**).

2.2.5 Quel protocole dans quel EVC3D ?

Un des aspects prépondérant dans les **EVC 3D** concerne la communication et la transmission des données lors de la collaboration. Que ce soit dans un **EV**, un jeu multijoueur ou un jeu sérieux (*serious game*), les participants interagissent en temps-réel ou

quasi-réel alors qu'ils sont situés à différents endroits géographiques. Le développement d'un **EVC 3D** nécessite donc des connaissances issues de plusieurs domaines comme la conception et l'implémentation de protocoles réseaux et de réseaux distribués.

Comme l'indique [Roberto *et al.*, 2014], le protocole le plus souvent rencontré dans les **EVC 3D** est l'**UDP**. Cette prépondérance s'explique d'après deux situations. Premièrement, dans des réseaux où l'accès est fiable car ces environnements n'ont pas (ou peu) de perte de paquet (ex : **LAN**). Deuxièmement, dans un contexte d'application où la perte de paquet n'est pas critique (ex : jeu vidéo, diffusion en continu de son, vidéo, **3D**...). **UDP** peut également s'utiliser dans les tout type de réseaux – **LAN** ou **WAN** (comme internet) –, car il est basé sur des datagrammes. Le fait qu'il n'effectue pas de vérification de délivrance lui donne un avantage quant à la taille des données envoyées dans les **EVCs**. La responsabilité de vérifier quels sont les paquets qui n'ont pas été délivrés et leur ordre d'arrivée incombe à l'application.

TCP est le protocole qui vient en seconde position. Les **EVC 3D** qui sont implémentés sur des réseaux haut-débit ne sont pas affectés par les étapes de vérifications que nécessitent une connexion **TCP**. D'après [Sung *et al.*, 2006], l'utilisation du protocole **TCP** sur internet dans le cadre des **EVC 3D** n'est pas recommandée à cause de la taille de l'en-tête et du ACK qui réduisent le débit des paquets. Dans ce contexte, le protocole **UDP** obtient de meilleurs résultats dans le cas où le flux de données est local (**LAN**). Les protocoles les plus adaptés pour communiquer dans un **EVC 3D** sont cependant **SCTP** et **RTP/RTCP**. Ils combinent les fonctionnalités de **TCP** et **UDP** et sont optimisés pour les applications multimédia (flux et temps-réel). **SCTP** offre la possibilité de choisir entre un système de livraison fiable (pour les paquets clés par exemple) ou non fiable (pour des mises à jour normales).

Depuis plusieurs décennies, les architectures **P2P** sont utilisées dans les **EVC**. Par exemple, l'assistance au rendu **3D** par les pairs pour le rendu est une méthode qui permet aux pairs hébergés sur des appareils avec des capacités limitées (bande passante, processeur, processeur graphique) de demander une partie du contenu de l'environnement aux autres pairs. Des systèmes récents comme celui de Martinez et al. [Martinez G. *et al.*, 2009] utilisent la localisation d'un utilisateur pour transmettre uniquement les mises à jours à son plus proche voisin. Ce système d'assistance par les pairs peut également être utilisé pour améliorer le rendu dans les environnements virtuels en ajustant la qualité du rendu en fonction du coût de calcul [Zhu *et al.*, 2011]. Koskela et al. [Koskela *et al.*, 2014] sont allés plus loin dans cette direction en proposant la méthode **RADE** (*Ressource-Aware P2P-assisted 3D Delivery Method*). **RADE** repose sur l'utilisation du **P2P** pour alléger le chargement et donc réduire le coût d'exploitation des

fournisseurs de service. Les clients peuvent également être inclus en tant que fournisseurs de services dans cette architecture. Le système utilise un serveur qui connaît la localisation des ressources et effectue le *load balancing* nécessaire pour distribuer les données en fonction des capacités des clients. L'impact énergétique d'un tel système a été évalué sur les mobiles ciblant trois optimisations possibles : la qualité visuelle, les performances et la consommation d'énergie.

Les architectures hybrides, combinant client-serveur et P2P sont également des solutions utilisées dans le cadre de l'apprentissage en ligne [Ekadiyanto et Hunger, 2012] et du BIM [Chen et Hou, 2014] par exemple.

En général, les systèmes P2P semblent n'être supportés que dans les environnements virtuels 3D. Ce phénomène est probablement dû au lien étroit entre la séparation spatiale de l'utilisateur et le besoin de distribuer les données à tous les participants.

2.3 Les systèmes distribués orientés événements pour la collaboration

2.3.1 Introduction

La plupart des systèmes basés événements sont très populaires lorsqu'il s'agit de collaboration[Helmer *et al.*, 2011]. Les événements peuvent être utilisé dans Computer-Supported Cooperative Work (CSCW) On retrouve par exemple la spécification d'événements composites pour le support de la collaboration dans la conception logicielle [Yuan *et al.*, 2002] ou encore la définition de patrons de conception dédiés à la collaboration dans les architectures basées événement [Verginadis *et al.*, 2009]. Dans ce dernier domaine Papageorgiou et al. [Papageorgiou *et al.*, 2011] proposent un assistant à la création de ces patrons dédiés à la collaboration. Ils s'appuient sur un système de recommandation basé sur le contexte qui utilise une représentation sémantique ([Web Ontology Language \(OWL\)](#)).

CoDesign est un exemple de [framework](#) permettant de faire de la conception logicielle de manière collaborative [Bang *et al.*, 2010]. L'utilisation d'une architecture basée événement permet à l'application d'être très extensible car très peu couplée. Chaque instance CoDesign intègre un intergiciel appelé CoWare en charge de la synchronisation de contenus édités de manière concurrente sur les différentes instances CoDesign, ainsi qu'un module chargé de notifier les architectes de situations de modélisation conflictuelles.

Eventuate est une plateforme qui se base sur un modèle de programmation événementielle ayant pour but de résoudre les problèmes liés à la gestion de données distribuées inhérents aux architectures microservices en utilisant **ES** et **CQRS**. Une de leurs applications exemple est un tableau Kanban collaboratif temps-réel construit sur leur plateforme¹².

2.3.2 Systèmes Publish-Subscribe

Le système **Publish-Subscribe (Pub / Sub)** est un mécanisme basé sur le paradigme des messages qui est beaucoup utilisé en Event Processing. Les *publishers* (ceux qui émettent des messages) et les *subscribers* (ceux qui souscrivent) sont couplés de manière lâche. Autrement dit, le *publisher* n'est pas forcément au courant de l'existence de ses *subscribers* et n'a donc pas de contrainte forte le liant à ces derniers. De son côté, le *subscriber* est souvent indifférent au *publisher* qui lui fournit les événements qui l'intéressent. La Figure 2.5 représente le modèle **Pub / Sub**. Le service de notification d'événement est utilisé comme passe-plat d'événements entre le publisher et les subscribers. Son rôle est de gérer les abonnements (souscription / désinscription) et de notifier les *subscribers* concernés par la publication d'un événement par un *publisher*.

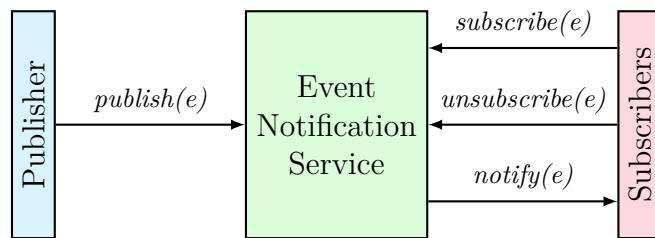


FIGURE 2.5 – Architecture Publish-Subscribe

Les modèles utilisant le paradigme **Pub / Sub** supportent naturellement une communication *many-to-many* entre les publieurs (*publishers*) et les souscripteurs (*subscribers*). De cette manière, le meilleur passage à l'échelle d'une application est facilité grâce à une topologie du réseau plus flexible.

Pour répondre à des besoins de mise à l'échelle, d'interopérabilité, de fiabilité, d'expressivité et d'utilisabilité, il s'appuie sur un modèle **Pub / Sub** basé types et attributs. En spécifiant d'abord le type et ensuite en filtrant sur les attributs des événements, ce modèle rend le routage des données intuitif pour le développement d'applications distribuées à grande échelle comme le commerce en ligne (*e-commerce*). Scribe [Castro *et al.*, 2002] et Hermes [Pietzuch et Bacon, 2002] sont des exemples de systèmes **Pub / Sub** basés sur les **Distributed Hash Tables(DHTs)**. Le premier se concentre sur la catégorie du sujet

12. <https://github.com/eventuate-examples/es-kanban-board>

(topic-based) tandis que le second supporte à la fois la communication orientée sujet et la communication orientée contenu (content-based) en utilisant un filtrage par agrégat. Les deux utilisent un nœud *rendezvous* pour chaque sujet ou type d'événement dans la couche réseau et construisent et maintiennent la distribution à partir de ce nœud *rendezvous*. TERA [Baldoni *et al.*, 2007] propose une dissémination sur deux couches orientée sujet. Le but est d'effectuer une distribution uniforme en fonction des intérêts des nœuds. La dissémination se déroule en deux phases. Pendant la première phase, un algorithme de marche aléatoire (succession de pas aléatoires dans le graphe) est utilisé sur la couche basse qui connecte les nœuds dans le but de localiser le nœud qui a souscrit au sujet. La seconde phase commence lorsqu'un *subscriber* est trouvé pour le connecter au *cluster* correspondant au sujet sur la couche haute. C'est là que l'événement est disséminé.

En juin 2017, Leonardo Quernozy présente une rétrospective des systèmes **Pub / Sub** en **P2P** massifs lors de la conférence DEBS'2017 à Barcelone. Les premières traces d'une telle architecture remontent à 1987 avec la publication de Birman et Joseph [Birman et Joseph, 1987] lorsqu'ils évoquent « the "News" service in the ISIS system » :

« *This service allows processes to enroll in a system-wide news facility. Each subscriber receives a copy of any messages having a "subject" for which it has enrolled in the order they were posted. Although modeled after net-news, the news service is an active entity that informs processes immediately on learning of an event about which they have expressed interest.* » [Birman et Joseph, 1987]

Après cette publication, les premiers systèmes **Pub / Sub** commencent à se développer. Le concept de routeur d'information est introduit par Oki et al. [Oki *et al.*, 1993] : un seul bus d'information pour tous les services, objets, données ... Les protocoles de communication ont une sémantique minimale, les objets (instances de classe) sont auto-descriptifs - i.e. ils sont capables d'introspection (service, opérations, attributs) pour adapter leur comportement au changement, les types peuvent être définis dynamiquement et la communication est anonyme. Puis des solutions plus avancées sont apparues avec le projet GRYPHON d'IBM [Banavar *et al.*, 1999], SIENA [Carzaniga *et al.*, 2000], REBECA [Parzy jegla *et al.*, 2010] et également REDS, JEDI, PADRES présentés plus en détails dans [Tarkoma, 2012]. Ces différentes solutions ont été conçues comme des systèmes de gestion avec des gestionnaires d'événements dédiées. Le passage à l'échelle est principalement effectué par un routage efficace des événements (comme le filtrage d'événements). Seuls quelques travaux ont utilisé le concept de *clustering* par intérêt comme moyen afin de réduire le nombre de notifications d'événement et éviter la surcharge. A partir des années 2000, les systèmes **P2P** ont commencé à émerger avec l'apparition de Gnutella et consorts,

les DHTs (Chord, Pastry, Kademlia) et Réseaux non-structurés (Cyclon, Scamp, ADH). Ces systèmes ont pour objectif de gérer des réseaux massifs, dynamiques (*churn*), résistants aux erreurs, et offrant des primitives de communications basiques. Les réseaux P2P promettent alors des propriétés intéressantes pour les infrastructures logicielles comme les systèmes Pub / Sub concernant le passage à l'échelle. Cependant, les systèmes Pub / Sub d'aujourd'hui reposent sur une connaissance totale du réseau pour être efficaces, du fait de leur taille limitée. La connaissance globale de l'information ne peut être collectée dans un système à grande échelle, c'est pourquoi la décentralisation de l'information est nécessaire. La connaissance peut vite devenir viciée dans un système dynamique décentralisé, ce qui implique de trouver une méthode efficace pour gérer la dynamité.

Les systèmes Pub / Sub ont donc l'avantage de proposer une EDA dont le fonctionnement s'adapte à plusieurs types de communication, que ce soit client serveur, P2P. Cette flexibilité a l'avantage de proposer un système à couplage lâche qu'il reste cependant difficile d'observer à grande échelle.

Outils de surveillance, contrôle de performances et validation ergonomique

Dans un système distribué comme Pub / Sub, les outils de comparaison sont souvent très hétérogènes et ne permettent pas forcément d'évaluer sur une même base la montée en charge. Carzaniga and Wolf [Carzaniga et Wolf, 2002] proposent quelques lignes directrices pour concevoir une suite de *benchmark* dans un système Pub / Sub, sans fournir de résultat spécifique, dont le but est double : la validation de l'interface et l'évaluation de la performance. La pertinence de l'interface a été définie par une série de questions/réponses posée au développeur. Elle adresse plusieurs aspects liés à l'Interface Utilisateur (IU) :

- le modèle de publication (structure, domaines de valeur, taille des objets) ;
- le modèle de souscription (portée de l'évaluation d'une publication reçue, type de langage, expressivité du langage pour la sélection) ;
- les méthodes d'accès à l'interface (accès local, mémoire partagée) ;
- la portabilité du système (support multi-plateforme) ;
- le type de service (fiabilité) et les fonctionnalités auxiliaires.

Pour évaluer la charge de travail d'un système distribué basé événements, Kounev et al. [Kounev et al., 2008] ont utilisé des techniques d'analyse opérationnelle pour caractériser le trafic du système et dériver une approximation de la moyenne des délais de livraison d'événements.

Beaucoup de recherches ont été menées dans le but d'améliorer les performances des systèmes collaboratifs utilisant une architecture distribuée. Bien que ces approches per-

mettent de passer à l'échelle de grandes quantités de données, cela requiert de lourds investissement pour installer et maintenir plusieurs serveurs dédiés. Une alternative à bas coût est d'utiliser les différents clients à disposition (qui sont en relation par le biais de la collaboration) pour avoir un traitement réparti des données sur la foule (*crowd computing*) [Li et al., 2015]. De ce fait, le contenu 3D est distribué par les pairs et non plus par le serveur. En combinant les ressources réseau du serveur et de plusieurs clients, on peut réaliser une distribution des données (événements) en améliorant les performances du système pendant les sessions de travail collaboratif.

2.3.3 Domain Driven Design

Domain Driven Design (DDD) ou Conception Pilotée par le Domaine est une approche de développement logiciel qui a pour objectif de définir une vision et un langage partagé (*ubiquitous language*) pour les personnes impliquées dans la construction d'une application [Evans, 2003]. Avec le DDD, un projet est vu à travers le prisme du domaine d'application qui le concerne (métier) et la logique du domaine en basant des conceptions complexes sur un modèle du domaine. Cela s'inscrit dans l'objectif d'initier une collaboration créative entre les experts techniques et les experts du domaine pour produire un modèle conceptuel – qui relève de problèmes spécifiques au domaine – raffiné itérativement. Les différents concepts du DDD sont listés en suivant :

Le contexte est le cadre dans lequel un mot apparaît qui détermine sa signification ;

Le domaine est une ontologie, une influence, ou une activité. L'étendue du sujet auquel l'utilisateur applique un programme est le domaine du logiciel ;

Le modèle est un système d'abstractions qui décrit les aspects sélectionnés du domaine et qui sont utilisés pour résoudre les problèmes liés à ce domaine ;

Le langage partagé est un langage structuré autour du modèle du domaine utilisé par tous les membres de l'équipe pour faire référence aux activités de l'équipe permettant d'éviter la redondance et les ambiguïtés dans un contexte donné.

Le DDD permet de connecter le modèle et son implémentation en offrant plusieurs avantages. La plasticité du système est mise en avant par l'expression de règles et de comportements qui facilitent les changements fréquents. L'accent mis sur l'identification des interactions dans le système encourage la mise en œuvre d'une interface orientée tâches (*task-based UI*). La testabilité fonctionnelle est intégrée via les règles métier explicitées et concentrées dans une couche spécifique de l'application. Cela les rend plus facilement identifiables et testables automatiquement. La robustesse est améliorée face aux changements dans le système d'information.

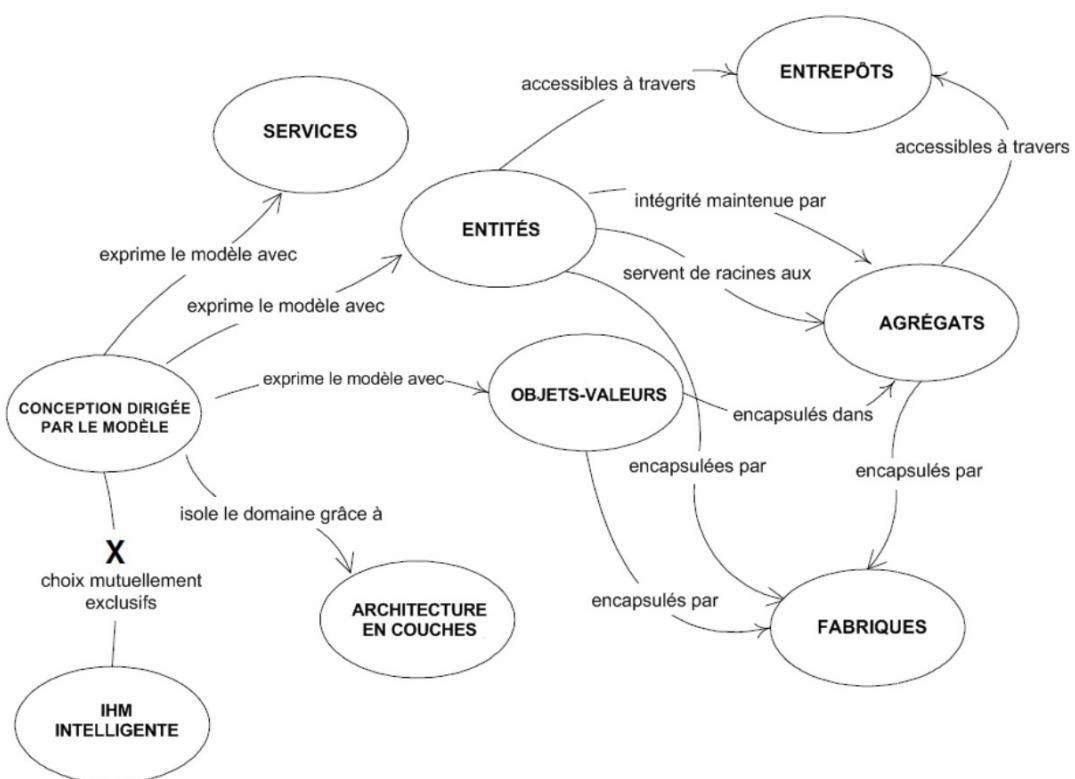


FIGURE 2.6 – Illustration du patron DDD et de ses artefacts (issue de [Avram et Marinescu, 2006] sous licence Creative Commons)

En DDD, il existe des artefacts qui permettent d'exprimer, créer et stocker un modèle lié à un domaine (voir Figure 2.6). Parmi les principaux se trouvent :

- **L'entité** (*entity*) : un objet qui n'est pas défini par ses attributs mais plutôt par une continuité et son identité. Par exemple, chaque scène possède des maillages qui utilisent des géométries. Si l'on considère un maillage comme unique dans chacune des scènes alors chaque maillage est une entité. Si on considère que ce sont les mêmes maillages qui sont réutilisés alors un maillage est considéré comme un objet-valeur.
- **L'objet-valeur** (*value object*) : un objet qui contient des attributs mais n'a pas d'identité conceptuelle. Il doit être traité comme un objet immuable. Par rapport à l'exemple précédent, on peut considérer que si les géométries sont réutilisées d'une scène à l'autre, ce sont des objets-valeurs qui ne seront jamais modifiés car les utilisateurs ne sont intéressés que par les informations qu'elles portent.
- **L'agrégat** (*aggregate*) : une collection d'objets qui sont liés ensemble par une entité commune (*root entity*), connue sous le nom d'agrégat souche (*aggregate root*). Ce dernier garantit la cohérence des modifications faites au sein de l'agrégat en interdisant aux objets externes de faire référence à ses membres. Par exemple, dans une modélisation 3D collaborative un utilisateur ne peut pas modifier le nom d'un autre utilisateur. Un maillage n'a également pas d'intérêt à connaître les informations des utilisateurs, c'est la scène qui gère l'interface entre les maillages et les utilisateurs.
- **L'événement du domaine** (*domain event*) : un événement auquel l'expert du domaine s'intéresse. Il est défini par un objet du domaine.
- **Le dépôt** (*repository*) : les méthodes pour récupérer les objets du domaine doivent être déléguées à un **dépôt** spécialisé afin de faciliter les implantations alternatives de stockage.

Les notions plus détaillées qui se rapportent au DDD sont présentées dans [Evans, 2003] et [Vernon, 2013]¹³.

Le DDD a également l'avantage de fonctionner en harmonie avec les principes de l'ES. L'approche logicielle DDD étant conçue pour refléter les événements se déroulant dans la réalité métier qui ne sont pas interchangeables – la plupart du temps. L'utilisation d'architectures basées événements est naturelle dans ce genre d'environnement.

Les disciplines liées à la 3D dans un contexte industriel ont besoin de pouvoir communiquer sur un langage partagé pour permettre à chacun des intervenants de s'exprimer

13. Les différents ouvrages publiés par Vernon s'attachent également à montrer la nécessité pour les différentes branches de l'informatique à proposer des systèmes d'information plus robustes et flexibles face aux nouvelles demandes.

dans la création du produit. Par exemple, la **CAO** est très utile dans un contexte d'ingénierie grâce à l'utilisation de quatre propriétés fondamentales telles que l'historique, les fonctionnalités, la paramétrisation et le haut niveau de contrainte.

2.3.4 Command Query Responsibility Segregation

Une solution architecturale courante en **DDD** contient quatre couches : l'interface utilisateur (présentation), la couche application (coordination de l'activité de l'application), la couche domaine (cœur du logiciel métier), la couche infrastructure (interface entre les couches, persistance des objets métier...). La Figure 2.7 montre que ces quatre couches s'accordent bien avec l'architecture **CQRS** qui sépare le flux d'écriture et de lecture dans le logiciel.

Introduit par Greg Young en 2009 [Young, 2009], **CQRS** est un patron de conception qui repose sur le principe de séparation des composants de traitement métier de l'information (écriture) et de la restitution de l'information (lecture). Le cadre offert par ce principe permet de lever certaines contraintes d'architecture comme le passage à l'échelle en faisant apparaître de nouvelles forces : la gestion de la concurrence dans la collaboration sur des règles métier propre à la modélisation **3D** dans des cadres d'application spécifiques. En effet, selon le type d'application, un ensemble de règles régit les droits concernant les types de modifications acceptables et qui est autorisé à le faire.

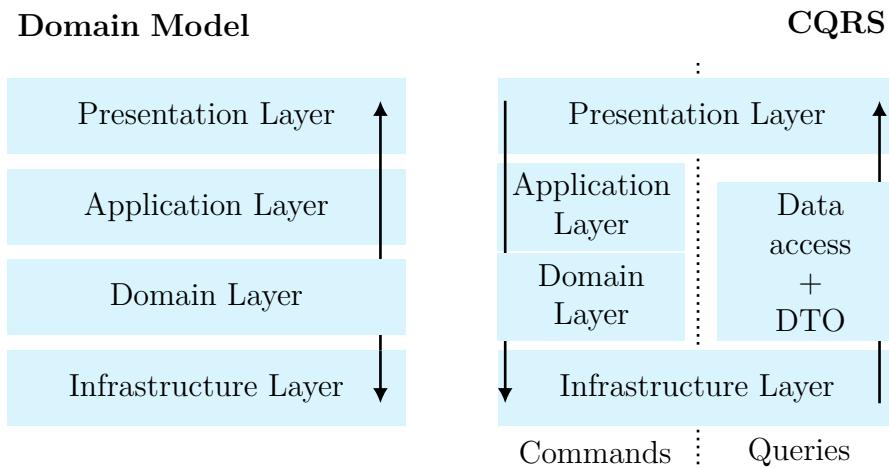


FIGURE 2.7 – Architecture en 4 couches du DDD (gauche) en miroir avec l'architecture CQRS (droite)

Le caractère vicié (*staleness*) d'une donnée dans un environnement collaboratif est récurrent. Une fois que la donnée a été montrée à un utilisateur, la même donnée peut

être changée par un autre utilisateur, elle est altérée. **CQRS** pallie cela en répondant aux besoins suivants :

- **En traitement / écriture** : besoins transactionnels, garantie de cohérence des données, de normalisation ;
- **En consultation / lecture** : dénormalisation, passage à l'échelle.

La plupart des architectures en couches ne font pas explicitement référence à ces problèmes. Le fait de tout sauvegarder dans une base de données centralisée peut être une étape dans la gestion de la collaboration, mais l'altération des données est souvent exacerbée par l'utilisation de caches comme accélérateur de performance. L'immuabilité en **CQRS** est un concept clé qui prévient la modification de l'état interne d'une commande ou d'un événement. Les commandes sont immuables car leur usage nécessite un envoi direct au domaine pour être traitées. Quant aux événements, ils sont immuables car ils représentent ce qui s'est produit dans le passé (qu'on ne peut donc pas changer).

2.3.5 Event Sourcing

Le Reactive Manifesto, apparu en 2014, est un document qui résume les propriétés clés liées aux systèmes distribués et encourage notamment le développement de systèmes « plus flexibles, à couplage faible et extensibles »[Bonér *et al.*, 2014] comme le **CQRS** et l'**ES**.

L'**Event Sourcing** (**ES**) est une approche complémentaire au **CQRS** pour gérer la concurrence des données et capturer l'intention de l'utilisateur. Ce patron de conception stocke les événements en mode ajout seulement (*append-only*) pour sauvegarder le résultat des commandes sur le domaine. Cela permet de conserver tous les changements qui ont mené à un état plutôt qu'uniquement l'état. Ce paradigme permet de recréer n'importe quel état d'un agrégat à partir de la liste d'événements qu'on lui a appliqué. Cette liste représente une base la vérité du système. **ES** permet de simplifier les tâches complexes dans des domaines complexes comme la **3D** en ne requérant pas la synchronisation de modèle de données et du métier.

L'**ES** est souvent présent dans des architecture asynchrones qui ont l'avantage de pouvoir utiliser des queues de message, plusieurs bases de données, et où la partie lecture est éventuellement consistante. La plupart de la littérature concernant ce patron se trouve en ligne, dans des billets de blog, des présentations ou de la documentation logicielle. La littérature académique est relativement réduite et se rapporte principalement à l'étude de l'évolution de graphes dans le temps [Erb, 2015, Erb *et al.*, 2017]. Cette section fournit

un aperçu des différentes définitions données de l'**ES** et du vocabulaire lié à ce patron de conception.

Martin Fowler a été le premier à utiliser le terme d'**Event Sourcing** en 2005. Il définit l'**ES** comme « une série de changements de l'état d'une application ». Il voit les événements comme immuables et le journal d'événement (*event log*) comme un stockage linéaire (*append only store*) des événements (Figure 2.8a). Les événements ne sont jamais supprimés, le seul moyen de l'"annuler" consiste à effectuer générer un événement rétroactif. Une fois qu'un événement rétroactif est ajouté, l'événement rétroactif agit à l'inverse de l'événement précédent pour compenser ses effets (Figure 2.8b). Dans ce billet, Fowler n'établit pas clairement la distinction entre les événements et les commandes qui déclenchent ces événements. Ce problème est considéré dans plusieurs travaux fondamentaux [Prakash et Knister, 1994, Sun, 2002, Weiss et al., 2009, Weiss et al., 2010], et approfondi par une revue de Chen et al. [Cheng et al., 2013].

Greg Young, important contributeur au domaine de l'**ES** (et **CQRS**), décrit l'**ES** comme « le stockage de l'état courant sous la forme d'une série d'événements et la reconstruction de l'état du système en rejouant cette série d'événements ». D'après lui, le journal d'événements a également un comportement linéaire : les événements qui sont déjà arrivés ne peuvent être défait. Ce que Fowler appelle événements rétroactifs, Young le décrit comme des actions inverses.

Udi Dahan Udi Dahan est également un auteur de billets de blog prolifique sur les systèmes **ES**. Dans sa définition de l'**ES**, Dahan insiste sur le fait que « l'état du modèle du domaine est persisté comme un *flux* d'événements plutôt qu'un simple instantané ».

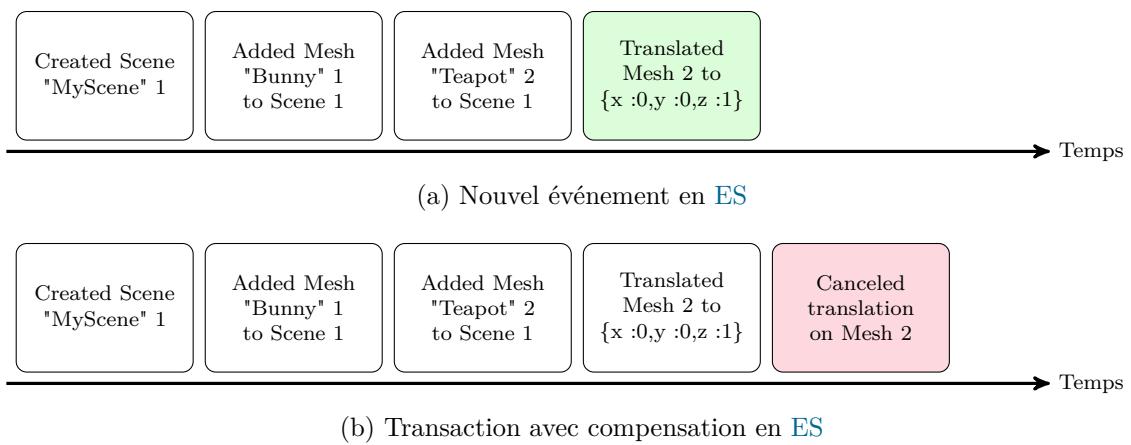


FIGURE 2.8 – Transaction en **ES**

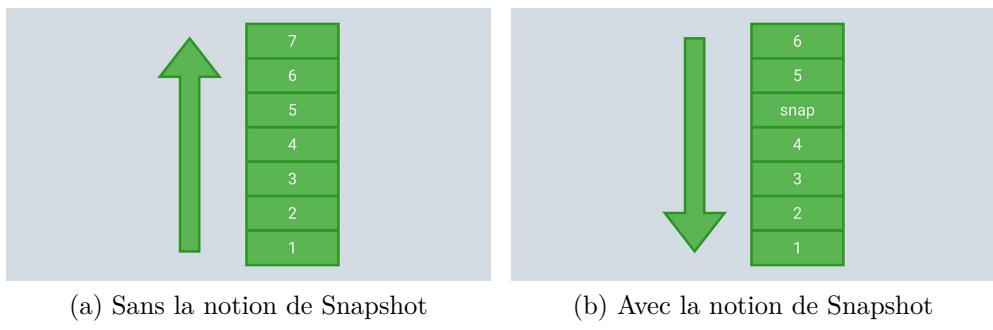


FIGURE 2.9 – Snapshot en ES

Les avantages et les inconvénients de l'approche **ES** dépendent du cadre dans lequel elle est utilisée. En reprenant ceux cités par Klamer [Klamer, 2013], ils peuvent être envisagés sous l'angle de la modélisation **3D** collaborative.

Avantages

L'**ES** peut apporter beaucoup d'avantages à une application, notamment lorsque les besoins en traçabilité de l'information sont importants comme dans la modélisation collaborative de données **3D**. L'historique du système est accessible tout au long de la vie de l'application, ce qui implique qu'il est non seulement possible d'accéder à l'état courant du système, mais également à toutes les actions ayant mené jusqu'à cet état. C'est un avantage certain pour les systèmes critiques, les applications d'Informatique Décisionnelle ou les applications collaboratives. La pratique la plus courante est de proposer un système principal et d'ajouter une multitude de sous-systèmes qui enregistrent les différentes métriques (analyse d'un ou plusieurs axes, *reporting* sur une propriété) du système principal. Avec l'**ES**, il est toujours possible de regarder « dans le passé » et de récupérer les données à partir de ce moment. Par comparaison, les avantages de l'**ES** sur l'Active Record sont :

- un journal complet de tous les changements d'état,
- une traçabilité et un débogage efficace,
- de très bonnes performances,
- pas de mapping objet-relationnel (ORM).

Exemple Une revue de projet d'une scène est effectuée dans le cadre d'une application collaborative de modélisation **3D** en ligne. Le chef de projet remarque qu'un objet a subi énormément de modifications par rapport aux autres. Il examine en particulier cet objet pour en connaître les causes. Voici deux exemples d'observations possibles : 1/ les spécifications ne sont pas assez claires 2/ deux collaborateurs sont opposés sur la façon

de modifier l'objet. L'origine de ces changements identifiée, le chef de projet pourra alors intervenir et clarifier le sujet.

Résolution Avec un système classique, la fonctionnalité doit être créée pour enregistrer ce traitement, puis être testée et implémentée (dans cet exemple, il en faudrait deux). Ensuite seulement, un rapport pourra être délivré. Dans un environnement utilisant l'[ES](#), tous les événements existent déjà donc les données sont déjà disponibles et ont seulement à être analysées (dans notre exemple, les informations sont inhérentes aux événements). De plus, les événements étant stockés depuis le début de la vie – mise en route – de l'application, beaucoup plus de données peuvent être analysées. Par exemple : demander toutes les modifications d'un objet depuis la dernière connexion d'un utilisateur pour lui communiquer visuellement les différences par rapport à sa dernière visite.

Inconvénients

En revanche, les performances de l'application peuvent être affectées par l'utilisation de l'[ES](#). En effet, pour récupérer l'état courant à partir de l'Event Store, il est nécessaire de le calculer à partir des événements reçus. A chaque événement créé (et stocké), ce processus sera alourdi car ils ne peuvent être supprimés. Plus la pile d'événements est longue, plus cela prend du temps (linéaire), considérant que les commandes ont parfois besoin de connaître une partie de l'état courant issu de ces événements pour être déclenchées.

Un [Snapshot](#) est une capture de l'état de l'agrégat à un moment qui correspond à l'empilement d'événements ayant mené à cet état. En créant un [Snapshot](#), on évite de reconstruire un état à partir du début de la vie de l'agrégat (Figure 2.9a) ; on empile les nouveaux événements à partir de ce snapshot (Figure 2.9b). L'utilisation du [Snapshot](#) est intéressante à partir du moment où la pile d'événements devient plus lourde que le [Snapshot](#) lui-même. Il est donc important de bien déterminer la périodicité du déclenchement du [Snapshot](#) (temporelle ou selon le nombre d'événements).

Le parcours des événements se fait classiquement du premier au dernier (*bottom-up*). Si on utilise les [Snapshot](#), on peut se permettre de les parcourir dans l'autre sens (*top-bottom*) jusqu'à trouver un [Snapshot](#) puis appliquer tous les événements qui sont arrivés entre ce [Snapshot](#) et l'état courant. Pour accéder à des données historiques plus anciennes que le dernier [Snapshot](#), le second parcours fonctionne aussi. Cependant, il doit être évitée pour ne pas créer de dépendance entre les [Snapshot](#). Sans les snapshots le modèle peut encore varier « librement » tant que l'on sait comment lui appliquer un événement passé. Le fait de travailler avec des [Snapshot](#) crée une dépendance des snapshots qui doivent

intégrer les modifications du domaine. Une solution est de recalculer les snapshots quand le domaine est modifié mais cela reste coûteux (et à éviter).

Event Sourcing versus Command Sourcing Les patrons de conception [Command Sourcing \(CS\)](#) et [ES](#) sont strictement déterministes pour avoir une exactitude rigoureuse de ce qui se passe dans le système.

Un événement représente quelque chose qui est arrivé dans le domaine. Un événement appliqué sur un état, donne un nouvel état. La condition pour appliquer un pur déterminisme en [ES](#) est la fonction suivante : $\text{État} \rightarrow \text{événement} \rightarrow \text{État}$. Cette fonction met en correspondance les (mêmes) entrées avec les (mêmes) sorties ; ce qui permet de se reposer dessus pour reconstituer un état à n'importe quel moment dans le temps. Le déterminisme est appliqué en assurant à l'événement tous les informations lui permettant de faire la transition (i.e. sans effet de bord). L'événement est alors considéré comme une encapsulation de toutes les informations pertinentes concernant la transition du système d'un état à l'autre.

Une commande va être déclenchée par l'utilisateur qui veut modifier le domaine. Une commande appliquée sur un état va produire un ou plusieurs événements (qui ne sont pas forcément appliqués à l'état courant de l'application par la suite) : $\text{État} \rightarrow \text{Commande} \rightarrow \text{Liste d'événements}$.

La différence principale entre un événement et une commande réside donc dans l'**intention**. D'un point de vue fonctionnel, le [CS](#) est un patron de conception lié à une décision qui produit plusieurs événements, tandis que l'[ES](#) se contente d'appliquer un changement à l'état courant. De plus, en [ES](#), l'événement stocké a été produit par l'agrégat. La différenciation majeure des deux patrons de conception s'accentue lors de l'interaction avec des systèmes externes. L'aspect fonctionnel de l'application de changement d'état de l'[ES](#) a l'avantage de permettre de reconstruire l'état sans effet de bord car elle ne travaille que sur l'état interne de l'application.

Historiquement, l'[ES](#) de Fowler dans ses premières versions ressemblait plus à du [CS](#). L'évolution de l'[ES](#) a conduit à revoir ce patron théoriquement sous une forme fonctionnelle pure, avec l'apparition, par exemple, de bases de données fonctionnelles comme EventStore ou de langages de programmation plus adaptés comme F#. Les deux patrons peuvent cohabiter si le [CS](#) a un cadre d'action bien délimité et que les composants de l'architecture ont un couplage lâche afin que seuls les événements produits par les agrégats ne modifient l'état interne de l'application.

2.4 Conclusion

La manipulation et la visualisation d'objets **3D** sur le web est une problématique qui peut se découper en différentes parties. La première concerne la modélisation collaborative **3D** sur le web. Les plateformes existantes essayent de répondre à plusieurs challenge : conserver la cohérence de l'environnement, proposer des solutions performantes et économiques (bande passante, énergie, graphique, stockage) pour s'adapter aux évolutions du web. L'un des aspects fondamentaux du web est l'accessibilité. Cela passe par la standardisation des technologies web par le **W3C**. L'émergence des standards comme WebGL (affichage **3D**) et WebRTC (connexion P2P entre navigateurs) a fait apparaître de nouvelles possibilités concernant la visualisation et le partage de données **3D**.

La seconde partie concerne la communication en temps réel au sein d'une application collaborative. Il existe plusieurs critères pour choisir le protocole de transport adapté à un **EVC 3D** : la fiabilité, l'ordonnancement et le risque d'erreur. Jusque là souvent cantonnés aux traditionnelles applications client-serveur sur web, les **EVCs** se montrent malgré tout intéressés par les architectures de communications P2P (éventuellement combinées à une architecture client-serveur). Dans ce cadre, les **EVC 3D** particulièrement lors d'édition collaborative de scènes **3D**, le choix d'une architecture **P2P** combiné à la haute fréquence des mises à jour, suggère le choix d'un mode non fiable, non ordonné et sans erreur. C'est l'application qui gère les paquets manquants et leur ré-ordonnancement pour favoriser la rapidité de la livraison.

La troisième partie s'intéresse aux propriétés des architectures orientées événements ou **EDA**. Le couplage lâche est une propriété qui s'accorde bien avec les systèmes distribués **P2P** ou hybrides. Les **EDAs** ont l'avantage de proposer une intégration de la partie métier transparente. Dans les différents systèmes de modélisation **3D** pour la visualisation ou la manipulation d'objets **3D** collaboratifs sur le web, cet aspect est souvent mis de côté. Or dans le cadre de la **CAO**, du **BIM** ou d'application au cadre industriel, ces points sont prépondérants. Différentes approches parmi les architectures orientée événements dans un système collaboratif permettent d'intégrer les règles liées au métier dans une application de modélisation collaborative. Dans un premier temps, l'architecture **Pub / Sub** propose un système de notifications pour les différents noeuds d'un réseau de manière lâche. De cette manière, la communication est homogénéisée et repose uniquement sur des notifications d'événement pour les entrées et sorties des clients. Le mécanisme d'abonnement autorise une grande variété de stratégies quant au routage des messages. Dans un second temps, différents patrons de conceptions sont présentés. Le **DDD** est plutôt considéré comme un patron de conception stratégique. Ses principes permettent de donner au logiciel une vision

orientée métier. A partir de ces orientations, **CQRS** vient proposer une discrimination entre la partie écriture (commande) et lecture (requête) dans l'application. Le passage à l'échelle côté lecture est favorisé grâce à la création de projections des données. Elles facilitent la restitution des données sans risques d'effets de bord. Le côté écriture permet de valider le contenu entrant selon les règles métiers du domaine. L'**ES** est le patron qui introduit la notion d'événement (« ce qui s'est passé »). L'état de l'application se reconstruit à partir de ces événements issus de l'expertise métier – déduite grâce au **DDD**. Les événements générés par **CQRS** sont stockés dans le journal des événements qui les partage aux autres collaborateurs, ce qui peut provoquer des conflits qu'il faut pouvoir détecter pour éviter les incohérences dans l'**EVC**.

Chapitre 3

Contributions scientifiques

Sommaire

3.1	Introduction	51
3.2	Modèle événementiel pour l'intégration du domaine 3D dans les EVC	52
3.2.1	Modèle général	54
3.2.2	Adaptation des patrons ES et CQRS	56
3.2.3	Cohérence éventuelle en CQRS	60
3.2.4	Potentielles applications et autres utilisations	61
3.2.5	Bilan	62
3.3	Architecture de communication hybride	63
3.3.1	Éléments constitutifs d'un pair	66
3.3.2	Architecture hybride « orientée états »	69
3.3.3	De l'état à l'événement	74
3.3.4	Architecture de communication hybride « orientée événements »	75
3.3.5	Comparaison des deux architectures	80
3.4	Conclusion du chapitre	80

3.1 Introduction

Les **EVC 3D** doivent considérer beaucoup de critères pour proposer un système réactif, robuste et permettant le passage à l'échelle en intégrant les contraintes liées à la gestion de données **3D**. La dimension collaborative ajoute les problématiques de gestion de données concurrentes. S'intéresser d'abord aux spécificités liées à l'assemblage d'objets **3D** dans une scène partagée permet de mieux comprendre les fonctionnalités spécifiques à la **3D** (transformations), à l'historisation et à la visualisation des données. De plus, les données générées doivent suivre un flux de validation avant d'être intégrées totalement au système. Les ressources nécessaires à la collaboration demandent une haute disponibilité et une forte réactivité qui implique la modélisation d'une architecture de communication répondant aux contraintes fonctionnelles (déttection de conflit) et technologiques (web).

Ce chapitre détaille les contributions scientifiques de cette thèse. Tout d'abord, il présente le modèle orienté événements pour la visualisation et la manipulation d'objets **3D** dans un environnement web. Cette première contribution insiste sur l'intégration de la partie métier de la **3D** par l'utilisation de patrons de conception (**DDD**, **CQRS**, **ES**). La réflexion concernant le découpage des opérations **3D** liées à la manipulation d'objets **3D** est d'abord expliquée ; puis, pour procurer plus d'autonomie aux utilisateurs, la contribution intègre le fait de déporter le patron **CQRS** uniquement sur le client afin qu'il soit capable de gérer la partie métier seul. La valeur ajoutée par le modèle orienté événements est l'intégration des aspects métiers de la modélisation 3D des actions des utilisateurs à la visualisation.

La seconde contribution de cette thèse concerne l'architecture de communication dans un environnement web. Celle-ci doit permettre au système d'être résilient, léger et transparent. L'architecture hybride proposée repose sur une combinaison de l'architecture client-serveur et l'architecture **P2P**. Cela tient à la nécessaire centralisation de l'information dans un contexte industriel et à la haute disponibilité des ressources que permettent les propriétés du **P2P**. Cette contribution est découpée en deux parties. La première expose une preuve de concept, sur la base du modèle présenté dans [Desprat *et al.*, 2015b], avec une architecture de communication simple proposant une diffusion des mises à jour par différentiel d'état. La seconde partie tire avantage de la mise en place du **framework** orienté événements présenté dans [Desprat *et al.*, 2016] et utilise la méthode de distribution des informations présenté dans [Desprat *et al.*, 2017] pour améliorer la résilience du système.

3.2 Modèle événementiel pour l'intégration du domaine 3D dans les EVC

La méthodologie orientée événements intègre plusieurs aspects souvent laissés de côté dans la littérature concernant le développement des environnements virtuels collaboratifs pour la visualisation et la manipulation d'objets 3D. Les événements peuvent, en effet, être utilisés dans le cadre de CSCW et d'EVC, qui ont évolué du simple collecticiel à de plus importantes organisations virtuelles comme on peut en retrouver dans des environnements scientifiques et d'ingénierie. L'idée de partager des ressources communes pour travailler sur un projet commun nécessite des outils propres qui implémentent des modèles de collaborations adaptés aux fonctionnalités spécifiques des systèmes distribués (forums, chats, tableaux blancs partagés ...). Ces collaborations sont routinières, les motifs de collaboration sont bien connus. Les motifs consistent en des segments de collaborations récurrents qui peuvent être capturés, encapsulés dans des composants distincts et réutilisés comme des solutions pour d'autres problèmes liés à la collaboration. Par exemple, les services de sensibilisation permettent de traiter les événements et de donner des informations à propos du travail collaboratif. En utilisant les événements suivis, les services d'analyse fournissent des statistiques à propos des collaborations passées et présentes.

Un système orienté événements a l'avantage d'être découplé. Dès lors, les données produites lors de la collaboration peuvent facilement être réutilisées pour un traitement secondaire comme la sensibilisation aux éléments de l'environnement. Le découpage de la modélisation logicielle et l'abstraction que le système apporte sont aussi l'occasion de s'intéresser à la façon dont sont générées les données produites par les utilisateurs et la façon dont elles sont affichées.

L'[Event Processing \(EP\)](#) est à la base de nombreuses applications informatiques dans les domaines de l'énergie, la santé, l'environnement, les transports, la finance, les services et l'industrie. L'[EP](#) réunit des méthodes et des outils pour filtrer, transformer et détecter des motifs dans des événements, dans le but de réagir à des conditions qui changent, généralement liées à des contraintes de temps [Chandy et al., 2011]. L'[EP](#) intègre plusieurs fonctionnalités pour :

- obtenir des données à partir de plusieurs sources en temps (quasi) réel;
- agréger et analyser ces données pour détecter des motifs qui indiquent la présence de situations critiques qui nécessitent une réponse;
- déterminer la réponse la plus adaptée à ces situations;
- et surveiller (*monitor*) l'exécution de cette réponse.

Le panorama d'applications et de technologies, proposé par [Hinze *et al.*, 2009], permet de définir les termes et les interprétations communes à différents domaines se basant sur le paradigme de l'[EP](#).

Constat

Par nature, une [EDA](#) est extrêmement peu couplée et hautement distribuée. Le créateur de l'événement sait seulement que l'événement se produit et n'a aucune idée du traitement que l'événement va subir par la suite ou de qui cela va concerner. C'est pourquoi les [EDAs](#) sont plus utilisées dans un contexte de flux d'information asynchrone. La traçabilité dans ces environnements devient alors un enjeu important. Facilitée par l'empreinte laissée par chaque événement, elle n'en demeure pas moins complexe selon l'échelle d'évaluation. À l'échelle d'un utilisateur, d'un groupe d'utilisateurs ou de plusieurs groupes, les acteurs restent des entités assez homogènes dans le cadre de la modélisation [3D](#) collaborative ce qui simplifie la tâche car le contexte et le domaine sont connus.

Le choix de baser la gestion des données sur le patron de conception [CQRS](#), combiné à celui de l'[ES](#), repose sur le constat suivant : dans un cadre industriel, le besoin de traçabilité de l'information est très important, pour suivre l'évolution d'un projet par exemple. Les [EDAs](#) reposent le plus souvent sur la communication client-serveur pour faciliter la gestion des données dans le système distribué. C'est pourquoi l'exploitation du patron de conception [CQRS](#) est traditionnellement développée sur la base d'une architecture client-serveur (pour récupérer les mises à jour). Ce fonctionnement ne permet pas le travail hors ligne.

Côté serveur, le stockage des données est de moins en moins cher : il est possible de stocker beaucoup de données de manière distante notamment grâce l'infonuagique. Le serveur a une puissance de calcul plus importante (et surtout ajustable).

Côté client, la puissance de calcul des machines sur lesquelles sont installés les navigateurs web évolue rapidement (notamment les appareils mobiles comme les *smartphones* et les tablettes). Les navigateurs suivent cette tendance en puisant dans ces ressources pour effectuer des traitements similaires à ceux que l'on trouve traditionnellement côté serveur et pour proposer des fonctionnalités avancées, telles que le stockage de données sur le client (IndexedDB, storageAPI), l'affichage [3D](#) (WebGL) et la communication en [P2P](#) ([WebRTC](#)). En déportant ainsi la charge que pourrait subir une architecture client-serveur côté client, les échanges réseaux sont limités car ils sont très coûteux d'un point de vue énergétique pour les appareils mobiles [Koskela *et al.*, 2015]. De plus, l'utilisation de la bande passante est onéreuse et parfois limitée - voire inexistante ; il est donc nécessaire de

tirer parti de tous les appareils participant à la collaboration, au lieu de tout faire reposer sur le serveur. Chaque appareil participant à la collaboration doit être autonome et le plus indépendant possible en termes de ressources (données, réseaux, validation experte).

Contribution Pour répondre à cette problématique, je propose un modèle pour la transmission des données liées à la **3D** et à la collaboration qui permet de limiter le nombre de requêtes et la taille des données transmises, sans perdre la traçabilité de celles-ci. L'idée est de profiter de la puissance du client pour créer une architecture assurant l'autonomie de l'utilisateur en cas de déconnexion volontaire (travail hors ligne) ou involontaire (coupe). L'utilisateur a la garantie d'avoir un historique performant qui profite de chaque connexion au réseau pour mettre à jour le système.

3.2.1 Modèle général

La mise en place d'une **EDA** pour faire de la modélisation **3D** engendre des avantages pour tous les métiers engagés (utilisateurs, développeurs, analystes métier). D'une part, la sensibilisation à l'historique des données et aux interactions inter-utilisateurs est partagée par les collaborateurs et les analystes métier. Les utilisateurs sont mieux informés de l'impact de leurs modifications et ont un aperçu général de l'évolution de la scène. D'autre part, la sensibilisation à la distribution des données est une composante importante pour les utilisateurs et les développeurs. En effet, la répartition de la charge permet de profiter du potentiel de calcul de toutes les parties prenantes du réseau.

Dans 3DEvent, le langage partagé se réfère au domaine de la manipulation d'objets **3D**, mais aussi à celui de la collaboration. Par exemple, le terme de maillage peut se référer à la fois au maillage géométrique ou bien au maillage de l'architecture réseau, d'où l'importance de définir les différents contextes en amont. Notons que le contexte de l'application peut faire varier les frontières d'un domaine. Le modèle issu du domaine défini permet de mettre en valeur les aspects métiers liés à l'application.

Spécification des événements dans le framework

Le framework étant orienté événements, cette section de thèse introduit la représentation et le vocabulaire utilisé dans la description des événements de 3DEvent. 3DEvent reprend la représentation proposée par Tominski [Tominski, 2006] illustrée par la Figure 3.1, et adapte ses propriétés pour la modélisation **3D** collaborative (visualisation et manipulation).

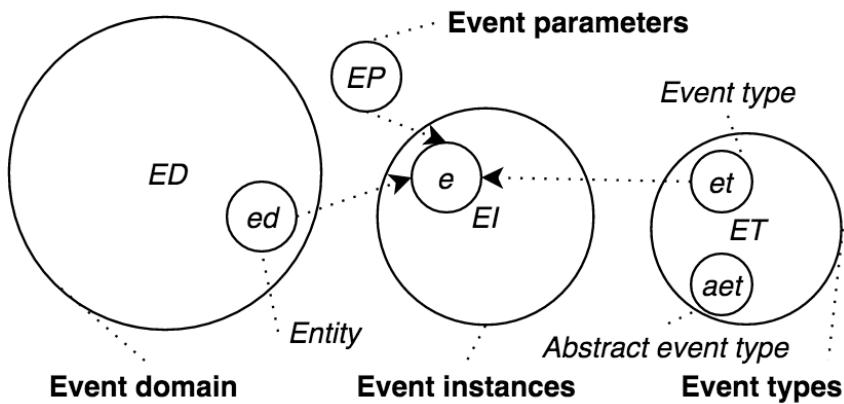


FIGURE 3.1 – Illustration des notions d'événements du domaine, des types d'événements, et des instances d'événements.

Pour décrire le cadre de référence dans lequel les événements se produisent, la notion de domaines d'événements est introduite. Un domaine d'événements *ED* (*Event Domain*) contient les événements qui respectent les types d'événements spécifiés. Par analogie avec le DDD, cet espace correspond au contexte borné (cadre d'action) d'un agrégat. À un agrégat, sont associés des types d'événements *ET* (*Event Type*) particuliers. Un *ET* est utilisé pour exprimer un intérêt concret envers les entités d'un *ED*. Les événements abstraits *aet* (*abstract event type*) sont compatibles avec l'*ED*. Une instance d'événements *e* – ou simplement événements – est composée du triplet *ED*, *ET* et *EP*. L'ensemble de toutes les instances d'événements possibles est exprimé par *EI*. *EP* (*Event Parameters*) correspond à l'ensemble des paramètres que peut avoir l'*EI*. Ainsi, tous les événements sont liés aux intérêts d'un *ED*. La spécification des événements nécessite de s'intéresser à tous les types d'événements qui sont utilisés lors de la manipulation et la visualisation d'objets 3D dans un *EVC*. La détection d'événements est un processus qui permet de récupérer les instances d'événements d'un *ET* en particulier. Cela peut être utilisé pour la visualisation de types particuliers par exemple. Dans 3DEvent, la spécification des événements est calquée sur les comportements observés lors des expérimentations des premiers travaux [Desprat *et al.*, 2015b, Desprat *et al.*, 2015a].

Le Tableau 3.1 présente les événements de l'agrégat Maillage – extrait du Tableau A.1 qui résume les différents types d'événements de base construits à partir du framework et les agrégats auxquels ils sont associés. Parmi ces événements, l'événement *meshAdded* et *meshDropped* semblent très proches, voire identiques quant au résultat que l'utilisateur peut obtenir. Cependant, la sémantique liée à chacun permet de différencier l'intention de l'utilisateur.

Tableau 3.1 – Événements de l'agrégat Maillage (extrait de Tab. A.1)

Événement	Dénomination	Description
Maillage ajouté (*)	meshAdded	Un maillage a été ajouté dans la Scène à partir d'une géométrie de la bibliothèque
Maillage déposé (*)	meshDropped	Un maillage a été déposé dans l'env. 3D de la Scène à partir d'une géométrie de la bibliothèque
Maillage supprimé	meshRemoved	Un maillage a été supprimé de la Scène
Maillage translaté	meshTranslated	Un maillage a subi une translation dans la Scène
Maillage pivoté	meshRotated	Un maillage a subi une rotation dans la Scène
Maillage mis à l'échelle	meshScaled	Un maillage a subi une homothétie dans la Scène

Ce tableau permet de donner un exemple qui reprend la description qui vient d'être présentée. L'*ED* correspond donc au « système de modélisation 3D collaborative ». Dans ce contexte, l'agrégat « Maillage » réagit à un certain nombre de types d'événements comme *meshAdded*, *meshDropped*... Les événements typés sont produits par l'agrégat en prenant en compte les paramètres nécessaires à leur instanciation. Par exemple, la production d'un événement *meshTranslated* prend en paramètres le vecteur de translation (x,y,z) et la référence de l'identifiant de la scène à laquelle il appartient.

Plus généralement, le domaine est la représentation des objets 3D d'un point de vue expert. Les objets du domaine représentent les données liées aux manipulations 3D sous une forme abstraite (géométrie, position), indépendamment des besoins du rendu (lumière, matériau...). Le framework intègre un générateur d'événements pour faciliter l'implantation de nouveaux types d'événements au plus proche des besoins des utilisateurs dans une application de modélisation 3D collaborative (appuyant la sémantique des manipulations).

3.2.2 Adaptation des patrons ES et CQRS

La Figure 3.2 montre le déroulement du cycle des opérations du modèle au sein de CQRS, de l'action utilisateur à la visualisation de son résultat. Du côté de la partie écriture (partie supérieure – *write part*), lorsque l'utilisateur déclenche une commande à partir de l'interface, la commande et ses paramètres sont récupérés et traités par le domaine pour être validés selon les règles métiers exprimées par ce dernier. Si la modification

est validée, le domaine produit ou modifie l'agrégat concerné. Ces modifications sont converties sous forme d'événements. Les événements sont ensuite transmis à l'Event Store où ils sont stockés avant d'être transférés à l'Event Publisher. L'Event Publisher joue le rôle d'interface entre la partie écriture et la partie lecture. Il est également responsable de la publication des événements sur le bus d'événements, Event Bus, où sont accrochées les différentes projections. Les projections sont nourries à partir des événements publiés auxquels elles sont abonnées. Enfin, une Vue (composant inclus dans l'IU) récupère les Data Transfer Objects(DTOs) contenant les mises à jour à partir d'une projection. La mise à jour d'une Vue peut se faire de manière passive – mode *push* – (ex. : mises à jour liées à la visualisation 3D des modifications des collaborateurs en temps réel) ou active – mode *pull* – (ex : aller sur une autre scène) selon le contexte.

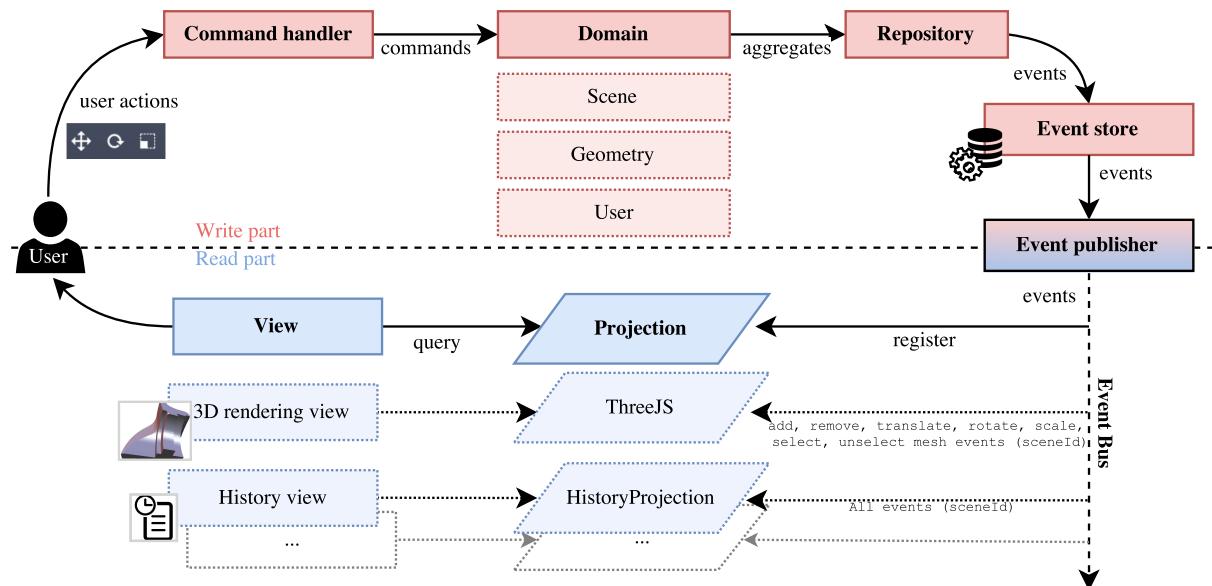


FIGURE 3.2 – Modèle de l'architecture client dans 3DEvent : la gestion du cycle de vie des données avec CQRS et ES dans un navigateur web, des actions des utilisateurs à la visualisation en passant par la synchronisation réseau. Issu de [Desprat *et al.*, 2017].

Dans ce modèle événementiel, la partie Commande permet de limiter l'introduction de données impropres dans le système via différents mécanismes. Chaque commande issue de l'IU entraîne la génération d'un ou plusieurs événements. Ces événements sont considérés comme « soumis » (*uncommitted*) mais pas encore « publiés » (*committed*) par le système. Une commande peut être rejetée pour des raisons variées. Ces raisons sont exprimées par différentes phases de validation de la commande. Une première validation se fait en amont de la création de la commande, avec par exemple une vérification de type des paramètres passés à la commande directement dans l'IU.

Le patron **Event Sourcing (ES)** permet de capturer tous les changements d'état d'une application sous la forme d'une séquence d'événements. Ces événements sont conservés dans un journal et peuvent être rejoués pour retrouver l'état de l'application. Les événements représentent des faits immuables qui sont ajoutés au journal les uns après les autres (sans pouvoir en être supprimés), ce qui permet des taux de transaction élevés et une réPLICATION efficace (cf Section 2.3.5). Dans 3DEvent, plusieurs composants d'**ES** sont étendus selon les applications :

- **Acteur** Un acteur consomme des événements à partir d'un journal d'événements et produit des événements pour le même journal d'événements. L'état interne dérivé à partir des événements consommés est un modèle d'écriture en mémoire (*in-memory*) et contribue à la partie commande (C) du CQRS.
- **Vue** Une vue est un acteur qui ne fait que consommer des événements à partir du journal d'événements. L'état interne dérivé à partir des événements consommés est un modèle de lecture en mémoire et contribue à la partie requête (Q) du CQRS.
- **Producteur** Un producteur est un acteur qui produit des événements à partir du journal d'événements pour mettre à jour la base de données. L'état interne dérivé à partir des événements consommés est un modèle de lecture en mémoire et contribue à la partie requête (Q) du CQRS.
- **Processeur** Un processeur est un acteur qui consomme des événements à partir d'un journal d'événements et produit les événements traités pour un autre journal d'événements. Les processeurs peuvent être utilisés pour connecter les journaux d'événements au traitement des événements..

Le journal d'événements Les événements produits par un des composants présenté ci-dessus peuvent être consommés par d'autres de ces abstractions s'ils partagent un journal d'événements local ou distribué. Un journal d'événements peut fonctionner sur un seul site ou être répliqué sur plusieurs sites. Le site est considéré comme une zone disponible qui accepte l'écriture d'un journal d'événements local, même s'il est partitionné sur plusieurs sites. Les journaux d'événements locaux, situés sur plusieurs sites, peuvent être connectés par le biais d'un journal d'événements dit « répliqué » (copié sur une autre réplique), qui a pour responsabilité de préserver l'ordre causal des événements. Les sites peuvent être situés à des endroits géographiquement distincts ou sur des nœuds à l'intérieur d'une même grappe (*cluster*), ou encore être sur le même nœud mais traités séparément selon les zones disponibles, qui sont nécessaires au fonctionnement de l'application. Les Acteurs et les Processeurs écrivent cependant toujours sur leur journal d'événements local. Les

composants peuvent soit collaborer sur un journal d'événements local sur le même site, ou au travers d'un journal répliqué sur différents sites. Il est important de différencier le journal d'événements de la base de données (côté serveur) ; la base de données peut ne contenir qu'une partie du journal. Une base de données peut également être considérée comme un élément complémentaire au journal d'événements, cependant et bien que parfois confondus, ils restent bien distincts conceptuellement. Le journal d'événements commun est la base des échanges pour communiquer par le biais d'événements de collaboration. Ce type d'architecture se retrouve dans différents cas d'utilisation :

- *Processus métier distribués.* Les acteurs de différents types utilisent des événements pour communiquer et parvenir à résoudre un problème commun. Bien qu'ils jouent des rôles différents dans le processus métier, ils réagissent à la réception d'événements (programmation réactive), en mettant à jour l'état de l'application et en produisant de nouveaux événements. Cette forme de collaboration est appelée collaboration dirigée par les événements.
- *RéPLICATION d'état d'Acteur.* Les acteurs de même type consomment les événements de chacun pour répliquer l'état interne avec une cohérence causale. Dans 3DEvent, les opérations concurrentes sont autorisées dans l'environnement pour mettre à jour l'état des acteurs répliqués et permettre la résolution interactive de conflits, en cas de mises à jour concurrentes et conflictuelles.
- *Agrégation d'événements.* Les vues et les producteurs agrègent des événements à partir d'autres composants pour générer des vues spécifiques à l'application. La collaboration événementielle apporte de la fiabilité quant à la gestion des données dans un système distribué. Par exemple, si un processus distribué échoue à cause d'un problème sur une partie du réseau, le système reprend automatiquement dès que les répliques sont à jour.

Les composants souscrivent à leur journal d'événements en s'accrochant au **bus d'événements**. Les événements nouveaux sont poussés vers les souscripteurs, ce qui leur permet de mettre à jour l'état de l'application avec une latence minimale. Un événement écrit à un endroit est publié de manière fiable auprès des souscripteurs sur ce site et des souscripteurs des sites distants. Par conséquent, les composants qui échangent par le biais d'un journal d'événements répliqué communiquent via un bus qui préserve l'ordre causal des événements de manière durable et tolérant au partitionnement. De ce fait, les services sur les partitions du réseau inter-sites (lien entre les sites) peuvent continuer à écrire des événements localement. La livraison des événements sur les sites distants reprend automatiquement lorsque les partitions sont à jour.

Le journal d'événements est local au site et fournit un ordre total des événements stockés. Le site est une zone de disponibilité qui héberge un ou plusieurs journaux d'événements. Les événements d'un journal d'événements sont répliqués sur les sites distants de manière asynchrone. Afin de lier des journaux d'événements (localisés sur différents sites) à un journal d'événements répliqué, les journaux d'événements locaux doivent être accessibles à partir des points d'entrées de réPLICATION. De plus, ces points d'entrée doivent être connectés entre eux afin de créer un réseau de réPLICATIONS. Un journal d'événements répliqué est représenté par un journal d'événements local sur chacun des sites participants.

cf section
p2p

Les points d'entrée permettent de gérer un ou plusieurs journaux d'événements. Ces journaux sont identifiés pour permettre à la réPLICATION de ne s'intéresser qu'aux journaux de même identifiant. Les journaux avec différents identifiants sont ainsi isolés les uns des autres et leur distribution peut donc varier selon les sites.

Les journaux répliqués fournissent l'ordre causal des événements stockés : l'ordre de stockage est le même sur tous les sites, ce qui veut dire que les consommateurs qui lisent les événements du journal local vont toujours voir les effets avant leurs causes.

3.2.3 Cohérence éventuelle en CQRS

La **Cohérence Éventuelle (CE)**, ou *eventual consistency*, propose dans un système distribué contenant plusieurs répliques d'avoir une coordination lâche entre ces répliques. Cela apporte de nombreux avantages en termes de disponibilité, tolérance aux fautes et sécurité des données, et évite l'intégration de protocoles comme *2 phase commit* ou de protocoles *Paxos* (*consensus*) complexifiant les échanges. La **CE** introduit l'idée que toutes les répliques se réconcilient au bout d'un moment (*forward progression*) pour avoir le même état final. Si le caractère vicié d'une information est détecté, le système doit le « réparer » pour obtenir le bon état. L'ordonnancement des événements durant les mises à jour reste identique lorsque les événements sont rejoués par la suite car l'ordre issu de l'**ES** est déterminé par l'ordonnancement des événements stockés localement. Au sein d'une réplique, tous les composants **CQRS** respectent cet ordonnancement. Les répliques distantes du journal d'événements sont cohérentes si les événements respecte l'ordre causale [Lamport, 1978] : les événements liés causalement ont le même ordre sur tous les sites, alors que les événements concurrents peuvent avoir un ordre différent. Cette propriété est importante pour obtenir une cohérence causale forte dans une application qui respecte le théorème **Consistency, Availability, and Partition Tolerance (CAP)** (Figure 3.3). « *The largest single benefit about CQRS is when you start running into problems with the CAP theorem* » [Young, 2010]. Young justifie ensuite que le lien entre **CAP** et **CQRS** est plus

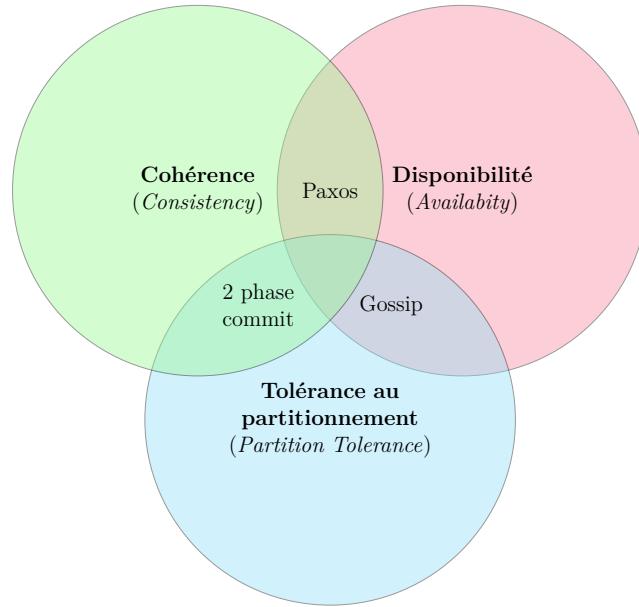


FIGURE 3.3 – Théorème CAP et les algorithmes de compromis

tenu qu'il n'y paraît. Même si **CQRS** ne permet pas d'éviter le dilemme de **CAP**, le fait que **CQRS** découpe le système en petites parties permet d'ajuster la cohérence séparément.

Dans **CQRS**, l'interaction avec plusieurs agents (utilisateurs, services) est découpée et subit un traitement réparti dont résulte la cohérence éventuelle de l'application. Dans la conception d'interaction pour l'interface, la synchronicité peut varier en fonction certains contextes. Les interactions asynchrones rajoutent souvent des étapes qui polluent l'interface. Par exemple, dans le cas de l'interaction suivante : **soumission de formulaire** → **envoi asynchrone** → **message de confirmation**, si l'utilisateur attend que la soumission soit une requête qui peut probablement échouer, alors l'asynchronisme se justifie pour être capable de fournir une explication à l'utilisateur en cas d'erreur. La probabilité d'échec peut être réduite en pré-validant la commande. Si personne d'autre que l'utilisateur ne travaille sur l'agrégat, la probabilité d'échec est quasi-nulle. De ce fait, l'asynchronicité rajoute une interaction inutile au flux dans le cas où peu de gens travaillent sur la même instance d'agrégat. Par exemple, il est intéressant de rendre l'exécution d'une commande synchrone et la mise à jour de la vue asynchrone. Les modèles de cohérence sont des décisions liées au métier car ils ont un impact direct sur l'expérience utilisateur.

3.2.4 Potentielles applications et autres utilisations

La conception et l'implémentation d'une plateforme comme 3DEvent qui est asynchrone, distribuée et orientée événements peut être appliquée à différents champs.

GIT-like app Les solutions pour faire de la gestion de version, comme MeshGit [Denning et Pellacini, 2018] par exemple qui fait du *diff and merge* de maillages polygonaux pour des données 3D, sont rarement implémentées (a fortiori en temps réel) sur des plateformes web à cause du coût et de la complexité que cela peut induire dans des architectures traditionnelles. 3DEvent peut reconstruire n'importe quel état antérieur grâce à son architecture orientée événements et indiquer les différences entre deux états.

Scénarios et *path recording* Pour des jeux sérieux, l'infographie 3D ou des études d'ergonomie, cette fonctionnalité est particulièrement pertinente. Le framework peut proposer une comparaison entre deux traces laissées par un ou plusieurs utilisateurs. Dans le cas où les utilisateurs ont la même tâche à réaliser, il est facile de faire la différence entre deux réalisations pour comparer, analyser et montrer les résultats, pour des raisons pédagogiques par exemple, ou pour relever des habitudes (de travail). Dans l'exemple du jeu sérieux, il est possible de comparer la trace utilisateur à la trace experte et de permettre de rejouer le même scénario plusieurs fois facilement pour observer l'évolution. Ce type de fonctionnalité est intrinsèque à 3DEvent.

Traçage utilisateur et *crowdsourcing* 3DEvent peut se révéler être un bon outil pour enregistrer la trace d'un utilisateur lorsqu'il navigue dans la scène. L'enregistrement du chemin de la caméra et des actions de l'utilisateur sous la forme d'événements sont des informations « issues de la foule » (*crowdsourcing*). En utilisant un processus d'apprentissage, il est possible de proposer de meilleurs chemins, repérer des points d'intérêt, ou même proposer des résumés de scène générés à partir des traces des collaborateurs (que s'est-il passé depuis la dernière connexion du collaborateur X ?).

Audit et outils de surveillance de données 3D L'ES fournit un mécanisme d'audit intégré qui assure la cohérence des données transactionnelles. Utiliser ce mécanisme pour faire un audit ou surveiller en temps réel l'activité de l'application peut fournir une meilleure compréhension du travail d'équipe, ainsi que de l'évolution de la conception. Cela peut permettre de repérer (avec du Complex Event Processing (CEP)), et corriger, certaines fonctionnalités afin d'améliorer l'ergonomie de l'application. Par exemple, si un événement est anormalement représenté dans le journal des événements, il sera possible de lever une alerte facilement.

3.2.5 Bilan

Le modèle orienté événements pour la modélisation 3D collaborative, présenté dans cette section, intègre les besoins métiers par différents aspects. Le suivi des directives liées au DDD permet de mieux cerner les règles métiers pour les intégrer dans CQRS.

La description de la typologie des événements manipulés insiste sur le compartimentage des objets métiers. La modélisation **3D** collaborative est minutieusement étudiée pour en dégager les quatre agrégats fondamentaux (la scène, le maillage, la géométrie et l'utilisateur) et les types d'événements qui leur sont associés. Cette étape introduit des différences fines sur les événements qui sont produits à partir d'un agrégat - un aspect intéressant notamment pour l'observation et la surveillance du contenu des sessions collaboratives, développées en Section 4.2.3. Les composants du patron **CQRS** favorisent le découplage de l'application pour obtenir une gestion de la cohérence plus fine.

La partie **ES** s'intéresse à la sauvegarde des événements dans un journal d'événements, et notamment dans le cas des applications distribuées en proposant un mécanisme de synchronisation pour détecter les conflits. La fonction principale de cette partie est de permettre de recréer un état de l'application cohérent en intégrant implicitement une gestion de version des événements.

Ce modèle introduit le cadre de travail pour utilisateur expert en manipulation **3D**. Il peut être facilement adapté pour différents types d'applications et étendu à d'autres utilisations. Pour mettre en avant l'expertise de chacun et profiter de toutes les ressources apportées par un utilisateur, une communication efficace de ces événements pour la collaboration doit désormais être établie entre les collaborateurs.

attention
garder
le bon
endroit

3.3 Architecture de communication hybride

Dans la section précédente, l'introduction du modèle orienté événements s'intéresse principalement à ce qui se passe au sein d'un seul client. Or la collaboration passe par la mise en relation des différents collaborateurs, mais aussi par l'échange des données nécessaires à la collaboration (données de connexion, mises à jour de la scène, sensibilisation...) Afin de pouvoir proposer un **SEC** adapté aux besoins de l'édition de scènes 3D, plusieurs types de granularité sont à distinguer :

- une granularité fine pour l'édition massive ;
- une granularité liée aux spécificités de la modélisation **3D** ;
- et une granularité reposant sur des technologies web (réseau, interface, interaction **3D**).

Les systèmes **P2P** qui sont orientés événements consistent en plusieurs noeuds interconnectés dont les fonctions et l'exécution des tâches sont similaires. Les pairs partagent directement leurs ressources (contenu, CPU, stockage, bande passante) sans nécessiter de serveur central. A la place, les pairs coopèrent au moyen d'événements qui prennent la

forme de messages lorsqu'ils se les échangent. Les systèmes P2P sont capables de s'adapter aux dysfonctionnements et à la dynamicité des pairs, tout en maintenant des performances acceptables. Les systèmes P2P sont généralement utilisés comme soutien aux services de l'application pour la communication et la collaboration, le calcul distribué, la distribution de contenus et pour les services d'intergiciel comme le routage et la localisation, ou encore l'anonymat et la confidentialité.

Constat Les SEC ont connu un fort développement avec l'intérêt croissant pour le P2P dans les années 2000. La base théorique des SEC s'appuie sur les propriétés de convergence, préservation de la causalité et préservation de l'intention du modèle CCI [Sun *et al.*, 1998]. La plupart des travaux liés aux SEC P2P s'intéressent à l'édition collaborative massive de documents textuels dont les propriétés de commutativité sont plus faciles à gérer (insertion / suppression) en comparaison à celles concernant la 3D (multiplication de matrices de rotation). La génération de conflits en 3D peut vite devenir incontrôlable. Il est donc nécessaire de mettre en place des mécanismes de détection de conflits et de maintenir une cohérence au cours des sessions de collaboration.

Contribution La mise en place d'une architecture de communication hybride permet de concilier à la fois les avantages d'une architecture client-serveur et ceux du P2P, en évitant certains désavantages occasionnés dans ces derniers (engorgement du serveur, pair seul sur le réseau...). Le terme « hybride » invoque un compromis entre la centralisation de l'information nécessaire à la prise de décision globale et la décentralisation des échanges utile à l'amélioration du partage de contenus 3D à l'échelle locale. En agissant sur ces deux échelles, la granularité de la collaboration est plus fine. Par exemple, le passage à l'échelle est facilité par la présence de ressources locales et la coordination des utilisateurs se fait à une échelle globale, ce qui permet également l'accès à une source de vérité centralisée (base de données) et commune. Dans le contexte des EVCs 3D, le but est d'obtenir une architecture de communication :

- entièrement basée web ;
- robuste face à l'évolution du nombre de collaborateurs ;
- efficace en termes d'accès et de distribution de la donnée ;
- et qui s'adapte à l'échelle selon les besoins de la collaboration.

Tous les travaux liés à cette thèse [Desprat *et al.*, 2015b, Desprat *et al.*, 2015a, Desprat *et al.*, 2016, Desprat *et al.*, 2017] s'appuient sur une architecture de communication hybride tripartite

(Figure 3.4) pour faire de la modélisation 3D collaborative : serveur, persistance et les pairs¹.

Les contributions portées par Desprat et al. dans [Desprat et al., 2015b] et [Desprat et al., 2015a] s'appuient sur des pairs ayant un rôle simple et proposent une transmission des mises à jour en tant que différentiel d'état. Les conflits lors d'opérations concurrentes sont évités grâce à un mécanisme de verrouillage. Plus tard, dans [Desprat et al., 2016], les auteurs mettent en place le modèle orienté événements qui impose l'évolution, au niveau réseau, du protocole de transmission (celui-ci reposant sur des notifications d'événements et la gestion des flux d'agrégat). Plus récemment, dans [Desprat et al., 2017], est apparue la distinction entre les deux rôles parmi les pairs : la partie serveur se compose désormais de pairs qui ont un rôle de vérification de la cohérence et de relais des données dans la couche P2P. Cette contribution intervient dans le but d'encourager la résilience du système et la vitesse de dissémination des informations dans le réseau, en favorisant la prompte détection de conflits (source de vérité plus proche des pairs producteurs). Cette extension repose sur la présence d'un journal d'événements partagé et répliqué sur les pairs participants à la collaboration.

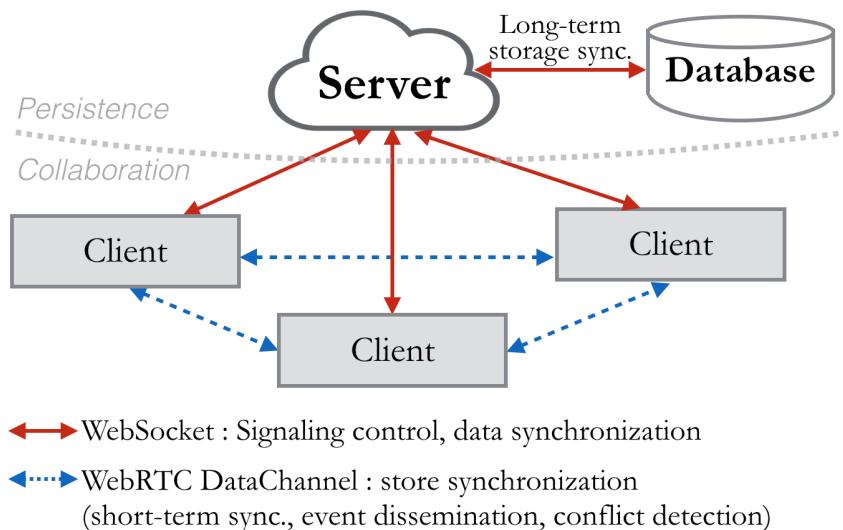


FIGURE 3.4 – Architecture hybride tripartite : serveur, persistance et les pairs.

Cette section décrit le modèle d'architecture mis en place pour gérer la transmission de contenu entre les différents clients qui participent à l'édition collaborative d'un espace de travail 3D partagé. Les différents protocoles de transmission, issus des travaux sus-nommés, sont présentés. Ils montrent ainsi l'évolution de l'architecture de communication de sa version naïve à la version étendue, plus proche des considérations métiers et offrant plus de liberté dans la création grâce la mise en place d'un système inspiré des CRDTs.

1. Les pairs sont appelés « Clients » dans les premiers travaux

Les différents éléments constitutifs d'un pair sont détaillés avant la présentation des deux types de pairs rencontrés dans l'architecture. Sont expliqués ensuite les mécanismes de mise en relation de ces composants pour que les différents participants d'une session collaborative puissent communiquer de manière transparente, cohérente et résiliente.

3.3.1 Éléments constitutifs d'un pair

Dans un réseau P2P, chaque pair est considéré comme un système à part entière qui contient ses propres problématiques d'implémentation. Avant de présenter la contribution liée à l'intericiel dans 3DEvent, il est nécessaire d'introduire les composants et les fonctions clés qui constituent un pair du point de vue strictement réseau en présentant une vue minimaliste d'un pair dédié au partage de contenu [Buford *et al.*, 2009, p.135-136]. Les fonctions sont divisées en trois :

- la couche de routage et d'échange de messages ;
- la recherche et le stockage de contenu ;
- ainsi que la configuration et sélection du rôle du pair.

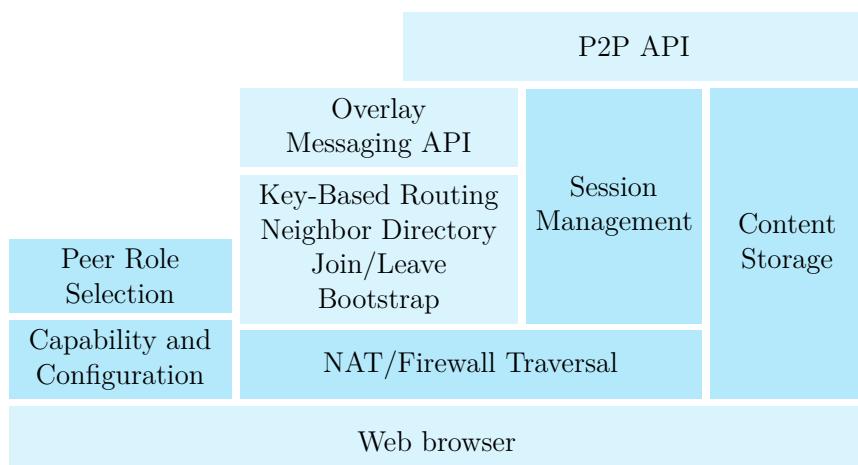


FIGURE 3.5 – Composants de chaque pair dans 3DEvent (point de vue réseau)

Chaque pair générique fournit de base une **API** pour les fonctions d'échange de messages et de recherche. L'**API** de recherche est principalement en lien avec le gestionnaire d'instances qui se charge de la recherche, mais il est possible de se connecter à des pairs via d'autres pairs si le serveur est absent ou pour renforcer le maillage. L'élément responsable de la couche de routage et d'échange de messages permet à chaque pair de maintenir un état de connexion avec ses voisins dans la couche ; maintenir cet état inclut la découverte de voisins et la maintenance à jour des états des voisins. Un pair possède un mécanisme d'amorçage (*bootstrap*) qui permet de localiser les autres pairs qui lui permettront de

rejoindre la couche P2P. Les étapes pour rejoindre ou quitter la couche sont proposées par le protocole d'établissement de connexion (cf signaling mechanism). La couche applicative permet d'échanger des messages avec les autres pairs dans le réseau en utilisant l'API réseau, et les messages reçus peuvent être transférés à leur destination en utilisant la fonction de transfert de messages.

ref section
signaling

Le contenu partagé est stocké localement pour pouvoir être accessible par d'autres pairs, ainsi que l'interface de l'application du pair courant. Le contenu doit être organisé pour la recherche d'informations afin de fournir une indexation et une interface de requête. L'espace de stockage pour les données partagées va correspondre aux différents états des agrégats [Desprat *et al.*, 2015b, Desprat *et al.*, 2015a] ou au journal d'événements [Desprat *et al.*, 2016, Desprat *et al.*, 2017].

La dernière fonction concerne la manière dont le pair auto-organise, à la fois, ses ressources locales et son rôle dans le réseau. Le pair détermine les ressources du système et du réseau disponibles et peut surveiller les changements éventuels régulièrement. Le pair a connaissance du rôle qui lui est attribué à sa création dans le réseau. Les rôles des pairs (super pair, relai, producteur...) dépendent de capacités comme les ressources du système et du réseau. Le rôle induit les capacités et la configuration de chaque pair, i.e. un pair actif est configuré uniquement pour produire, stocker et relayer les données tandis qu'un pair passif ne fait que stocker et relayer les données. La stabilité du pair peut être indiquée par son niveau de *churn* (dynamicité : rejoindre et quitter un réseau) par exemple. La Figure 3.5 peut être affinée en fonction du type de réseaux et d'applications. Chaque pair comprend un protocole pour la gestion de sessions (ex : REST API) et pour l'échange de données (ex : RTCDataChannels) pour pouvoir participer à la collaboration.

Les architectures de communication des contributions se divisent en deux parties, chacune correspondant au paradigme qu'elles utilisent respectivement : état ou événement. Selon le paradigme auquel elles font référence, elles peuvent être décrites par plusieurs critères :

revoir

Spécificité des composants de l'architecture Les pairs sont la base de la dissémination de l'information. La partie serveur centralise les données liées à la persistance long terme et décentralise la réPLICATION des données à court terme. Cela évite au serveur d'être le point central des échanges en déchargeant les canaux passant par le serveur.

Gestion de la session La gestion de la session suit le cycle de vie des pairs au sein du réseau lorsqu'ils participent à une session collaborative. Pour intégrer le réseau P2P

et participer à l'édition d'une scène **3D** de manière collaborative, les clients passent par les étapes suivantes :

1. Phase de *signaling*
 - (a) Signification de la présence auprès du serveur de *signaling*
 - (b) Récupération de la liste des identifiants des pairs auxquels se connecter
 - (c) Création d'une connexion **P2P** avec les autres pairs
 - (d) Souscription aux flux
2. Phase de synchronisation
 - (a) Requête des données manquantes
 - (b) Récupération des données manquantes
 - (c) Ré-ordonnancement des messages reçus
 - (d) Maintien de la cohérence (détection de conflit)
3. Phase de génération de nouvelles données valides
 - (a) Publication des données aux voisins
4. Phase d'abandon
 - (a) Notification aux voisins
 - (b) Fermeture des canaux

Ces quatre phases sont distinctes même si, lorsqu'un pair initialise ou rejoint une session collaborative (séquence d'initialisation), les phases **1**, **2** et **3** s'enchainent pour ce pair. Durant la session collaborative, les phases **2** et **3** s'alternent. La phase **4** se produit lorsque qu'un client quitte une scène de manière volontaire (changement de scène, fermeture de l'onglet du navigateur) ou involontaire (fausse manipulation, navigateur en panne, coupure internet) signifiant la fin de la session collaborative.

Gestion du stockage et structure des données Le stockage peut prendre plusieurs formes : *in-memory*, stockage local, disque dur...selon le type de stockage adapté et disponible. Dans l'architecture hybride, il existe deux niveaux de stockage : la persistence long-terme, distante – la base de données – et la persistence court-terme, locale – le navigateur. Le serveur assure, d'une part, la centralisation du stockage à long-terme, et d'autre part, la mise en relation des différents clients.

Gestion de la synchronisation et de la cohérence (détection des conflits) Il existe différentes possibilités de synchronisation des pairs.

L'édition concurrente lors de la collaboration peut être gérée de différentes manières, selon si elle est plutôt pessimiste ou optimiste lors de la réPLICATION des données

(voir Section 2.1.1). L'intégration d'une solution ou l'autre a de lourds impacts sur l'expérience utilisateur (fonctionnalités, interface utilisateur) et la gestion de la cohérence dans le réseau P2P.

Protocole d'échange La couche P2P fournit une dissémination rapide de l'information et

finir

3.3.2 Architecture hybride « orientée états »

Les premiers travaux liés à cette thèse [Desprat *et al.*, 2015b, Desprat *et al.*, 2015a] se sont intéressés principalement à la mise en place de l'architecture de communication en basant les structures de données sur des états (plutôt que des événements). Cette section décrit les choix faits lors de la modélisation de l'architecture « orientée état », pour le stockage et le transfert par exemple, en lien fort avec les technologies web sous-jacentes.

Spécificité des composants de l'architecture

Le système présenté repose sur une architecture REST (REpresentational State Transfer) concernant les échanges clients-serveur. Cette architecture permet de séparer les responsabilités entre le client et le serveur en séparant l'IU du stockage. REpresentational State Transfer (REST) propose une interface uniforme. Cela implique que chaque ressource est identifiée et manipulable par des représentations définies (modification, suppression).

Chaque pair correspond à un client (navigateur web) qui contient l'intergiciel P2P offrant des fonctionnalités limitées, l'interface utilisateur contenant l'environnement 3D et un espace de stockage reposant local (IndexedDB).

La persistance long terme est une base de données NoSQL. Ce type de bases de données a tendance à être optimisée en lecture (grâce à un langage de requête dédié) et permet d'éviter les structures rigides. Cette flexibilité est utile lors du stockage des modèles 3D et encourage la conservation d'autres métadonnées (ex : relations entre les différents objets) pour faciliter la traçabilité. Avec une base de données centralisée, le système possède une source de données fiable et autoritaire qui permet à un utilisateur de récupérer le bon contenu quand il revient sur une scène. La totalité du document, contenant la scène, est envoyé à l'utilisateur lorsqu'il y accède.

Gestion du stockage et structure de données

Avec une base de données NoSQL utilisant des schémas dynamiques, les données sont stockées sous forme de document. Chaque document est auto-descriptif et peut contenir

des valeurs qui s'imbriquent sous la forme d'une structure en arbre hiérarchique. Une collection consiste à grouper des documents, équivalent dans une base de données relationnelle à la notion de table. La base de données contient deux types de collections : les scènes et les géométries. Une collection de scènes contient des scènes qui sont décrites par leur identifiant et les méta-données de l'espace virtuel **3D**, ainsi que la liste des contributeurs et la liste des maillages. Cette liste correspond à une association entre un identifiant de maillage, les méta-données de l'objet et l'identifiant d'une géométrie existante. Les géométries sont, quant à elles, stockées avec leur identifiant propre et l'objet 3D complet donné sous le format **JavaScript Object Notation (JSON)**.

Sur chaque pair, il existe un stockage relationnel (IndexedDB) pour chaque objet de la base de donnée récupérée. Cela permet au client de faire des requêtes directement dans son espace local si besoin, par exemple en ajoutant un maillage. C'est également une source pour transférer ses modifications aux autres clients. Les navigateurs permettent désormais de stocker des quantités de données importantes localement avec une durée de vie illimitée. Les paramètres des opérations fondamentales effectuées sur la base de données (**Create, Read, Update, Delete (CRUD)**) sur les collections sont très bien supportés par les requêtes REST.

Gestion de la session

Lorsqu'un utilisateur rejoint une session, il suit le déroulement des opérations décrit dans la Figure 3.6. Dans un premier temps, afin de récupérer les données liées à l'espace de travail, le client web de l'utilisateur qui se rend sur une scène récupère tous les objets associés à la scène dans la base de données. Les objets de la scène sont renvoyés à l'éditeur pour les afficher. Dans un second temps, le système établit les connexions P2P entre les clients de manière automatique et complète (topologie réseau en maillage complet) après avoir récupéré les identifiants des autres utilisateurs auprès du serveur de *signaling*. Une fois la connexion établie, chaque collaborateur voit son environnement **3D** dans l'éditeur mis à jour, indiquant la position du nouvel arrivant. Les collaborateurs éditent ensuite la scène **3D** qui produit, sur les différents objets, des requêtes **CRUD** – incluant les opérations de translation, rotation et homothétie – destinées au serveur (qui transmet les modifications à la base de données) puis aux collaborateurs. Enfin, dans un dernier temps, lorsque l'utilisateur quitte la scène, il en informe le serveur de *signaling* qui s'occupe d'indiquer aux clients restants l'abandon du client pour qu'ils puissent mettre à jour leur interface en conséquence.

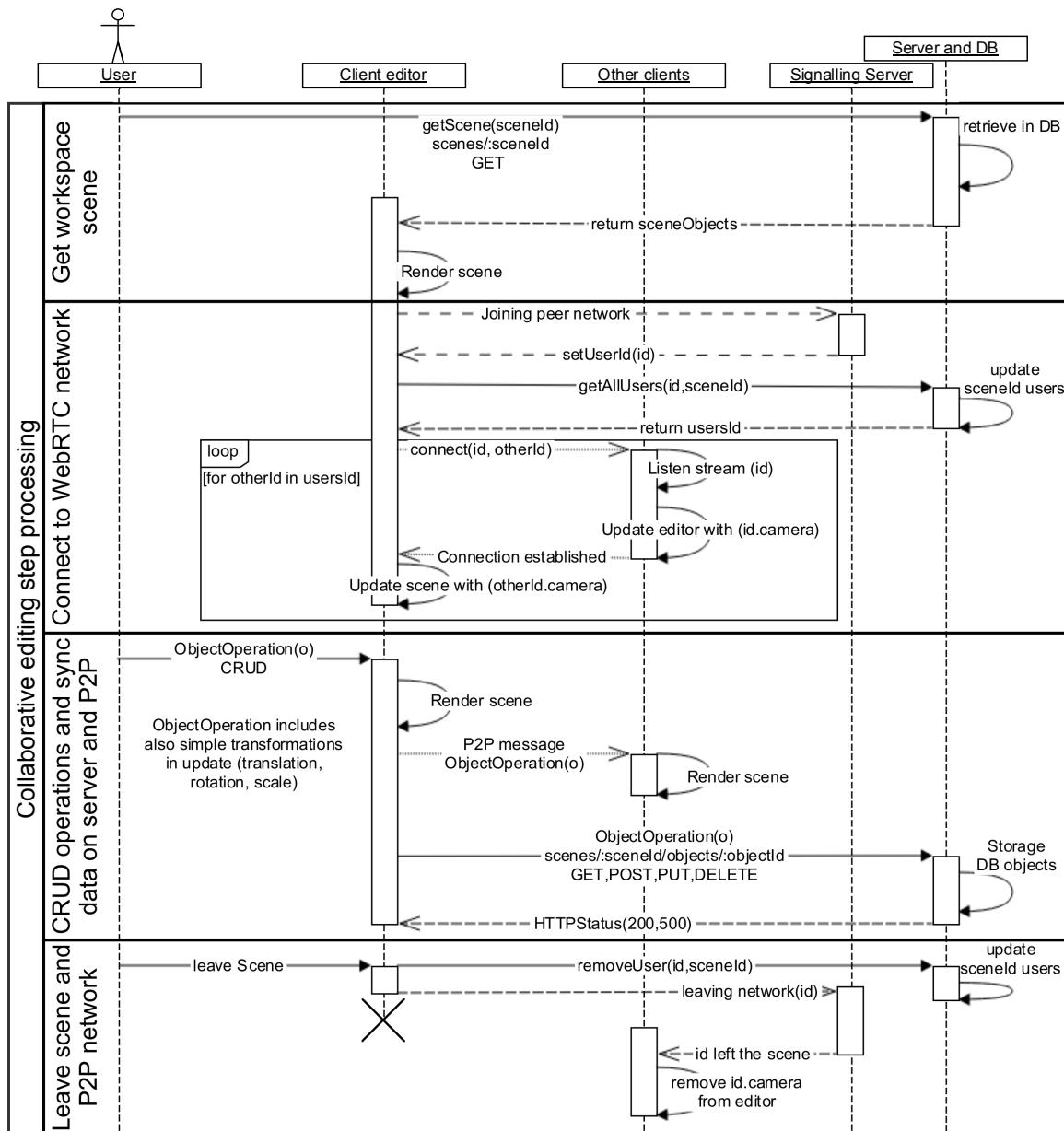


FIGURE 3.6 – Diagramme de séquence de la gestion de la session dans une architecture « orientée état »

Gestion de la synchronisation et de la cohérence

Les travaux [Desprat *et al.*, 2015b] et [Desprat *et al.*, 2015a] présentent une version naïve du processus de synchronisation. La cohérence est garantie par un système de verrouillage des objets. Cela évite notamment les sélections concurrentes et par conséquent les éditions concurrentes. Lors de la récupération de la scène, elle est considérée comme cohérente. Ensuite, le choix d'avoir des connexions P2P ordonnées et fiables, ainsi qu'un

maillage de pair complet, implique que toute donnée envoyée par un pair est forcément reçue par les autres.

Lors de la connexion d'un nouveau client, a lieu la synchronisation des deux systèmes de persistance pour obtenir les mises à jour :

1. depuis le client, où l'on distingue trois cas :

- (a) travail "*offline*" (hors ligne) : l'utilisateur a travaillé hors ligne et doit maintenant publier "en ligne" son travail. Les mises à jour sont publiées sur la base de données ; le serveur vérifie si aucun conflit ne survient puis fusionne (*merge*) les nouvelles entrées avec l'existant ;
- (b) travail "*serverless*" (en collaboration avec entre pairs sans le serveur) : dans le cas où le serveur est absent, les clients peuvent continuer de créer en collaborant. Ces données n'étant stockées que sur le client, il est nécessaire de les transmettre à la base de données dès qu'une connexion est possible. Cela peut être fait en une fois ou de manière partagée ;
- (c) travail "*online*" (en collaboration entre pairs comprenant la partie serveur) : le client envoie régulièrement ses nouvelles modifications pour qu'elles soient intégrées à la base de données.

2. depuis la base de données : Le client reçoit toutes les nouvelles mises à jour de la scène depuis la dernière fois qu'il s'est connecté. Cela peut également inclure des mises à jour qui sont en conflit avec ce qu'il y a dans son propre espace de stockage, qu'il lui faut donc modifier. Dans le cas où un utilisateur est seul connecté, la base de données est la seule source disponible pour la mise à jour du client.

Le fait de synchroniser un client avec une base de données NoSQL dès que cela est possible permet de supporter les connexions intermittentes comme pour les appareils mobiles dans cette architecture ; cette approche est connue sous le nom de « *offline first* » [Gadea *et al.*, 2016].

Topologie et protocole d'échange

Cette architecture est basée sur une topologie de maillage complet (*full mesh topology*) pour le réseau **P2P**. Les échanges se font donc à deux niveaux : entre les pairs, ainsi qu'entre chaque pair et la base de données. Pour le premier niveau, la couche réseau **P2P** est assez naïve dans le sens où les connexions sont considérées comme fiables et ordonnées ; le client n'a alors qu'à publier toutes les modifications qu'il effectue et les envoyer à tous les clients (auxquels il est nécessairement connecté). Les pairs échangent des différentiels

d'état sur les objets. Ces messages ne contiennent pas d'information sémantique sur le type d'action effectuée par chaque utilisateur.

Discussion des problématiques liées à une architecture de communication orientée « états »

La base de données centralisée est beaucoup sollicitée lors des sessions collaboratives, ce qui peut mener à une surcharge similaire aux architectures uniquement client-serveur. En combinant client-serveur et P2P, la charge aurait dû être réduite et transférée aux autres pairs, notamment lors de la phase de récupération d'une nouvelle scène qui repose entièrement sur le serveur. Chaque pair assume la responsabilité dans l'envoi de ses modifications à tous les autres pairs. Cette tâche ne profite pas du réseau P2P de manière optimale car la topologie ne profite pas de la fonction relais des pairs. Cela pourrait alléger la distribution dans la couche P2P et responsabiliser un peu plus les pairs pour faciliter le passage à l'échelle. En effet, la topologie en maillage complet limite le passage à l'échelle car le nombre de connexions va croître de manière exponentielle.

Par ailleurs, les requêtes CRUD ne sont pas très expressives concernant le métier. Aucune vérification n'est donc effectuée sur la validité des données transmises dans cette architecture. De plus, il peut arriver, du fait de latences réseaux que les modifications des différents utilisateurs s'entrelacent à l'arrivée et ne rendent pas l'intention de l'utilisateur. Cela peut mener à des dé-synchronisations fortes.

Concernant la gestion de la concurrence, l'utilisation d'une solution pessimiste dans un environnement P2P peut parfois mener à des inter-blocages si un utilisateur quitte la scène sans avoir relâché l'objet, c'est à dire sans en avoir informé la base de données et les autres utilisateurs. Même si l'application peut prendre le relais et permettre la relâche du verrou au bout d'un certain temps, l'expérience utilisateur peut être dégradée pendant cette période. Le maintien de la cohérence est à peu près garanti dans ce modèle si l'on considère un environnement où la fréquence des modifications n'est pas très élevée, et que tous les clients communiquent leur horloge locale pour établir un référentiel de temps concernant les mises à jour.

Enfin, le transfert par différentiel d'état utilisé pour réduire la taille des données liées au changement d'état des objets lors de la transmission des données peut varier grandement selon le type de changement effectué, rendant peu prévisible la charge des canaux de communication.

3.3.3 De l'état à l'événement

Afin d'intégrer la notion d'historique dans une application web, plusieurs indicateurs sont nécessaires (version, granularité de l'historisation, type et taille des données manipulées). Cette section décrit le processus pour passer d'un système basé états à un système basé événements dans l'intérêt de réduire la somme des données transmises sans perdre d'information concernant la logique métier.

Description des variables Une scène S contient l'ensemble des n objets x ($S = \{x_0, x_1, \dots, x_n\}$). Chaque élément de S est associé à une taille calculée comme la somme des modifications m selon la version v .

Pour tout objet x appartenant à la scène S on associe une taille $w \in \mathbb{R}$. w correspond à la taille totale des données transmises pour modifier l'objet à chaque version v ($v=0$ correspond à la taille de l'objet à l'état original, i.e. l'[Event Store](#) est vide). La taille totale de la scène, notée S_w , correspond à sommer la taille de tous ses objets (équation 3.1). Toutes les variables évoquées sont exprimées en octet pour avoir des termes avec des unités homogènes.

$$\text{Taille d'une scène } S : S_w = \sum_{i=0}^n w_i \quad (3.1)$$

Transmission par état Dans un système où à chaque modification on transmet l'état complet (équation 3.2), la taille des éléments transmis correspond à la somme de tous les états par lesquels l'objet est passé. Cette valeur va augmenter par paliers de tailles du même ordre de grandeur. La transmission par état propose un historique issu d'une sauvegarde chronologique. Cependant elle est lourde voire redondante et n'offre pas d'information sur l'intention de l'utilisateur (quelle opération a permis d'arriver à cet état ?).

$$\text{Par état : } w_i = \sum_{v=0}^n state_v \quad (3.2)$$

Transmission par différentiel d'état Dans un système où l'on transmet un différentiel d'état (équation 3.3), la différence entre deux états peut être difficile à exprimer car il s'agit d'un objet 3D dont les informations, stockées dans un fichier, ne sont pas linéaires. Le différentiel peut beaucoup varier jusqu'à atteindre une taille proche de l'état actuel si une transformation génère beaucoup de modifications par rapport à l'état précédent. A contrario, une modification sur la position d'un objet peut être très légère. La fonction représentant la taille des ressources transmises augmentera par palier avec au pire

les propriétés de la proposition précédente (équation 3.2). Le différentiel d'état permet d'alléger la transmission par rapport à la proposition précédente, ne contenant cependant pas d'information sémantique sur ce qui est transmis et les opérations qui ont conduit à l'état d'un objet 3D de la scène.

$$\text{Par différentiel d'état : } w_i = state_0 + \sum_{v=1}^n (\underbrace{state_v - state_{v-1}}_{\text{differential}}) \quad (3.3)$$

Transmission par événement La transmission par événement (équation 3.4) repose sur le principe du différentiel d'état (*differential*) exprimé ci-dessus (équation 3.3) en ajoutant la notion de type d'événement (*eventType*). Le type d'un événement indique le nom de la transformation à appliquer pour passer d'un état à l'autre. Un événement contient par nature une sémantique «métier» associée à chaque opération qui donne des indications sur l'intention de l'utilisateur.

$$\text{Par évènement : } w_i = state_0 + \sum_{v=1}^n (\underbrace{eventType + differential}_{\text{event}}) \quad (3.4)$$

Dans un **EVC 3D**, il y a des événements concernant à la fois la manipulation 3D, la navigation et les éléments se rapportant à la collaboration (par exemple : prendre le point de vue d'un collaborateur). La taille d'un événement est variable (comme un différentiel d'état) avec un minimum contenant le type d'événement (*eventType*) dû à sa structure type/paramètres. Cette proposition est un peu plus lourde à transmettre que la précédente mais ajoute une valeur sémantique à l'historique.

3.3.4 Architecture de communication hybride « orientée événements »

Cette architecture est une combinaison des contributions présentées dans [Desprat *et al.*, 2016, Desprat *et al.*, 2017] : elle repose sur les éléments du framework orienté événements (Section) qui sont également en relation avec la couche réseau (Figure 3.7).

ref mo-
dèle event

Spécificité des composants de l'architecture

L'Event Store est un composant clé dans le traitement des événements. Présent sur chaque pair, il prend en entrée des événements de deux natures : ceux qu'il a générés via la partie commande (internes) et ceux reçus via le réseau principalement (externes).

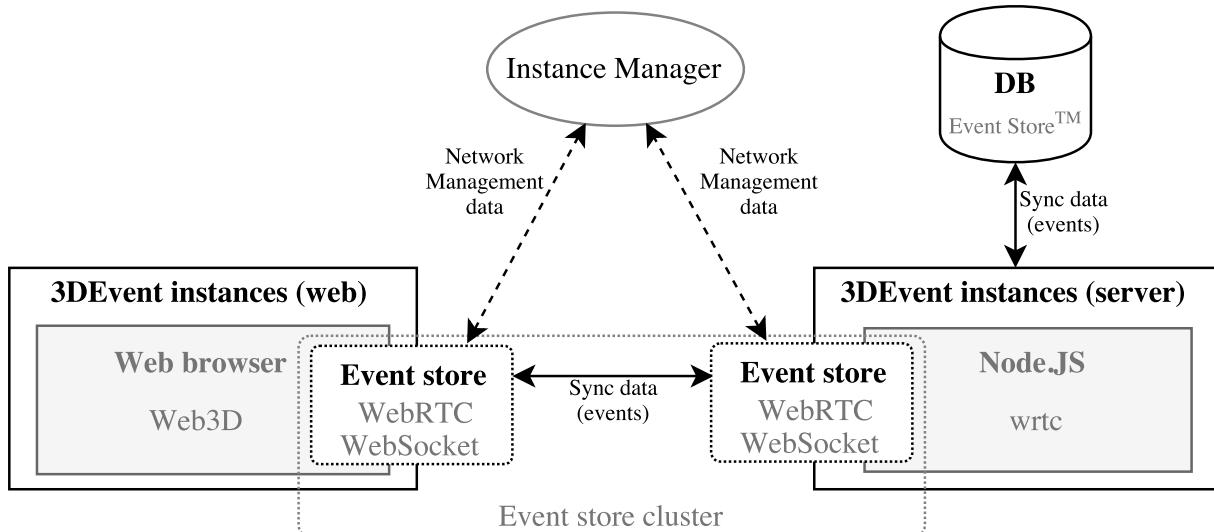


FIGURE 3.7 – Architecture de communication « orientée événements »

L'Event Store produit en sortie des événements dit « cohérents » qui peuvent être publiés par la suite sur l'Event Publisher. Un Event Store est composé de deux types d'éléments :

- l'**Event Stream Manager (ESM)** qui gère les flux d'événements : structure de données présente dans chaque Event Store qui doit permettre l'accès en lecture et en écriture des agrégats qu'elle contient. Un **ESM** contient des flux d'événements ordonnés temporellement pour chaque agrégat géré ;
- les **Network Bridges (NBs)** qui servent de connecteurs réseaux : responsables de la gestion d'une connexion P2P, i.e. gère le flux de données entrant et sortant vers chaque pair auquel il est relié.

La Figure 3.7 montre des pairs (appelés instances) de différentes natures :

- Instance Web : produit, stocke et relaie des événements aux autres instances.
- Instance Serveur : stocke et relaie des événements aux autres instances et à la base de données.

En plus de servir de relais comme une Instance Serveur, une Instance Web est productrice d'événements. C'est en général le type d'instance par lequel un utilisateur accédera à l'application. Les Instances Serveurs, en comparaison avec l'architecture présentée précédemment, sont des « serveurs » qui participent directement à la couche réseau. En cela ils aident à la dissémination des données et peuvent rapidement être montés pour garantir une disponibilité des données auprès des autres instances (notamment en cas de panne). Toutes les instances sont coordonnées par l'**Instance Manager (IM)** qui est responsable de mettre les pairs en relation. Il est notamment le serveur responsable dans le mécanisme de *signalling*.

L'ensemble des Event Stores forme une grappe (*cluster*) qui gère le journal d'événements partagé de manière répartie entre toutes les instances.

Gestion du stockage

Local avec l'ESM Lorsque l'Event Store reçoit de nouveaux événements, l'**ESM** crée ou récupère le flux d'événements associé à l'agrégat dans un premier temps. Puis stocke l'événement à la suite de ceux présents dans le flux de l'agrégat.

Long-terme et distant avec une base de données fonctionnelle Dans un contexte industriel de collaboration, l'information doit être disponible sur le long-terme, facilement accessible par l'entreprise. La persistance à long-terme stocke le journal d'événements qui est la source de vérité de l'application. Elle peut également stocker des projections prédefinies, calculées à la volée ou encore des *snapshots* de l'application.

Sur la base du modèle orienté événements, les données à stocker sont les événements. Ils sont immuables et sont des données purement fonctionnelles. L'utilisation d'une base de données fonctionnelle permet de se reposer sur l'immuabilité des événements. L'argument souvent opposé à l'utilisation de telles bases est le coût de l'espace de stockage. Or le coût de la redondance et de la non localité du traitement des données a chuté au cours de ces dernières années. Une base de données dont les données muent – i.e où chaque donnée peut être modifiée à n'importe quel moment – ne permet pas de conserver l'historique des modifications (ex : *Active Record*). Dans une base de données dite fonctionnelle, la structure de données doit elle-même être fonctionnelle et donc immuable – i.e. ne peut être modifiée a posteriori et fait seulement référence à d'autres données immuables. Une telle base de données est « une interface vers des *snapshots* versionnés » [Méric de Bellefon, 2012].

Dans le cas de l'**ES** les événements sont considérés comme des deltas (avec quelques métadonnées supplémentaires) sur les agrégats. C'est donc sous leur forme originale qu'ils sont stockés. Cela évite également les transformations de données (et la perte d'information ou l'ajout de complexité) qui sont nécessaires dans les **Object-Relational Mappings(ORMs)**, en lecture et en écriture.

voir section **ES vs CS**

Gestion de la session

Dans [Desprat *et al.*, 2017], lorsqu'un pair initialise ou rejoint, la séquence pour rejoindre une session collaborative évolue pour prendre en considération l'intégration des événements et la synchronisation des journaux d'événements. La Figure 3.8 représente la séquence d'actions nécessaire à une instance 3DEvent (*idA*) pour rejoindre le réseau

contenant déjà d'autres instances 3DEvent. L'action *join* est exécutée lorsqu'un utilisateur envoie ses informations de connexion sur un portail de connexion (à partir d'une instance web) ou lorsqu'une instance serveur est lancée. Cette action ajoute le nouveau pair à la liste des pairs présents sur le réseau. La liste est gérée par le gestionnaire d'instance, et retourne la liste des pairs avec lesquels le pair doit se connecter. Pour chaque pair *idB* de la liste renvoyée *ids*, *idA* utilise le mécanisme de signalisation (offre/demande). Le mécanisme est déclenché par l'instanciation d'un **NB** dans l'Event Store de *idA* puis celui de *idB*. Afin de resynchroniser les deux pairs (après cette série d'échanges asynchrones), *idA* et *idB* s'échangent des métadonnées sur la situation respective de leurs **ESM** afin de se synchroniser.

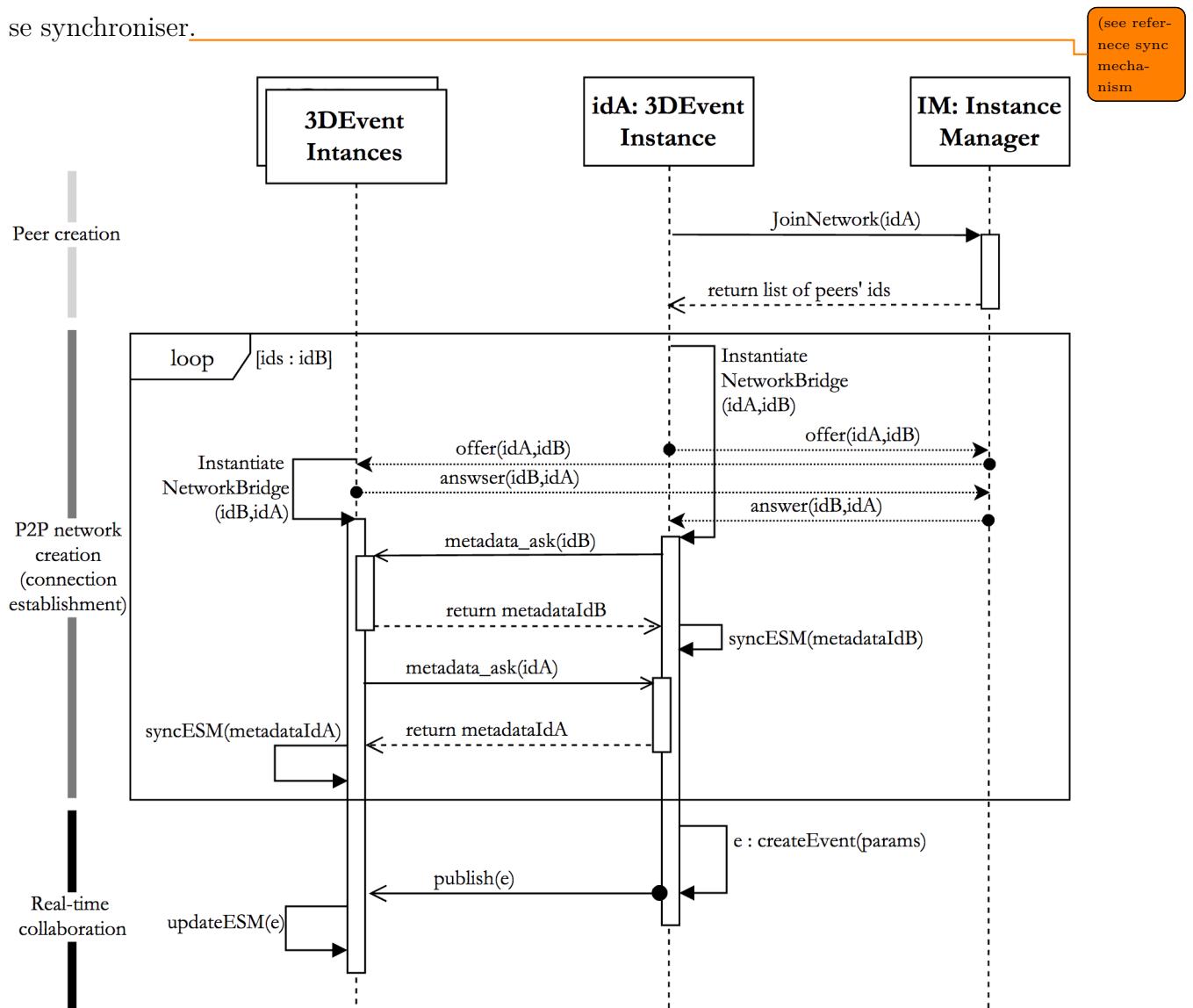


FIGURE 3.8 – Protocole de connexion au réseau d'instance 3DEvent

Gestion de la synchronisation et de la cohérence

Les événements sont considérés comme « cohérents » lorsqu'il n'y a pas d'erreur de cohérence dans l'agrégat, i.e. lorsqu'il n'y a pas de doublon dans les numéros de versions et qu'ils sont bien ordonnés. Lorsqu'un **ESM** ne rencontre pas de problème de cohérence, alors le dernier index correspond au numéro de version de l'agrégat.

Lorsque l'Event Store reçoit un événement interne, l'**ESM** récupère (ou crée) le flux d'événements associés à l'agrégat référencé par l'événement. La cohérence de la version est alors vérifiée en comparant la version attendue (exposée dans les méta-données de l'événement) et la version actuelle de l'agrégat. Si les deux numéros de version sont égaux, l'événement est ajouté à la fin du tableau du flux pour être stocké dans l'**ESM**, sinon une exception est levée. La gestion des exceptions est expliquée dans .

Mécanisme de gestion de version 3DEvent intègre une procédure de gestion de version dans l'**Event Store** afin de gérer au mieux la cohérence des données (voir Section). Pour être cohérent, l'agrégat concerné par ces événements doit produire une nouvelle version sans être en conflit avec la précédente. En passant la version attendue v_a au gestionnaire de conflit, on est à même de la comparer avec la version courante v_c . Il existe deux cas où un conflit est levé :

- a) La version v_a correspond à la valeur d'initialisation
- b) La version v_a est différente de la version v_c

Dans a), après une action, la version initiale de l'agrégat ne peut être la même. Cet item peut sembler évident mais il est important de le noter car il dépend entièrement de la valeur initiale choisie pour les agrégats du **framework** (on peut commencer à n'importe quelle valeur – 1, 0...).

ref section

parler de b), et redite par rapport a au dessus

Topologie et Protocole d'échange

L'architecture hybride orientée « événements » tire profit de la présence des Instances Serveur pour réduire la responsabilité des pairs producteurs (Instances Web) dans la distribution des données. En effet, les Instances Serveur qui participent au réseau permettent de proposer plus de points de distribution de données directement reliés à la base de données. De ce fait la politique de connectivité entre les pairs peut être réduite, la topologie du réseau correspond à un maillage partiel (*partial mesh topology*).

Discussion des problématiques liés à une architecture de communication orientée « événements »

Architecture réactive à l'attrition. Tous les éléments participants à la gestion de données, actifs dans la distribution ou producteurs de données, sont au même niveau. Le réseau P2P est renforcé par la présence de pairs destinés à faire le relais dans la distribution des données.

La gestion de la cohérence

finir

3.3.5 Comparaison des deux architectures

Tableau 3.2 – Récapitulatif des deux approches d'architecture de communication

Critère	HybridEvent	HybridState
Type stockage pair	Fonctionnel (event)	Relationnel
Type BDD	Fonctionnel (event)	NoSQL (Doc. JSON)
Cohérence	Éventuelle	Forte
Synchronisation	Push-pull	Push
Gestion de conflit	CRDT-like	Verrouillage des objets
	Plusieurs pairs sont liés	Lourde charge pour le
Robustesse	au serveur BDD : disponibilité ++	serveur lié à la BDD : disponibilité - -
Connectivité pair	One-to-many (policy)	One-to-all
Passage à l'échelle	Oui	Faible
Orientée métier	Oui	Non
Transmission	Notification événement	Diff. état
Topologie Maillage	Partiel	Complet
Type données	Évènement	État
Type requêtes	Projection / DTO	CRUD / REST
Historique des données	Oui	Non
Défaire / Refaire	Oui (compensation)	Oui (annulation commande avec effets de bord)

3.4 Conclusion du chapitre

Dans ce chapitre nous avons vu les différents composants de l'architecture de communication pour la réalisation d'un **EVC 3D**. En mettant en avant la technologie WebRTC, nous avons montré qu'il était possible de réaliser une architecture qui respecte les standards du web et l'interopérabilité nécessaire à ce type d'environnement. La mise en place d'une architecture décentralisée dans un environnement distribué permet de mettre à

contribution tous les acteurs de la collaboration. De ce fait, l'accessibilité des ressources est renforcée par la présence de nombreuses unités présentes sur le réseaux. Cela permet à la fois de récupérer du contenu rapidement et d'octroyer une autonomie certaine aux contributeurs.

Chapitre 4

Implantation

Sommaire

4.1	Introduction	83
4.2	3DEvent : Plateforme web de manipulation collaborative d'objets 3D	83
4.2.1	Éditeur 3DEvent	83
4.2.2	Interface utilisateur orientée tâche	84
4.2.3	Flexibilité de la visualisation	90
4.2.4	Bilan	90
4.3	Intergiciel P2P pour l'échange de données 3D pour le paradigme événementiel	91
4.3.1	Données d'échange	92
4.3.2	Synchronisation des données	93
4.4	Résumé des choix techniques	95
4.5	Bilan	97

4.1 Introduction

Cette section présente l'implantation de 3DEvent, la plateforme web de manipulation et de visualisation collaborative d'objets 3D réalisée à partir du modèle présenté dans le chapitre précédent.

La première partie s'intéresse à l'intégration du framework pour proposer une application d'assemblage d'objets 3D. La présentation commence avec la description de l'[IU](#) et des fonctionnalités proposées dont le principe de sélection fantôme et le système de visualisation flexible reposent sur le système de projection du modèle orienté événements. En proposant une application basée sur les standards du web, elle est accessible au plus grand nombre sans requérir d'installation et peut évoluer dans le temps avec les standards.

La seconde partie présente l'implémentation de l'architecture de communication avec les différentes implémentations des acteurs du système, les politiques de connexion et l'intergiciel [P2P](#) qui permettent l'échange de données entre les collaborateurs.

Enfin, la troisième section récapitule et discute les choix techniques pour l'interface utilisateur et la partie réseau qui ont été fait dans l'implémentation.

4.2 3DEvent : Plateforme web de manipulation collaborative d'objets 3D

Dans le chapitre précédent, une des contributions présentées correspond au modèle orienté événements : le [framework](#) 3DEvent. L'implantation d'un tel système contient plusieurs éléments comme l'éditeur web et son interface utilisateur. Les éléments de sensibilisation à la collaboration sont également implantés comme le mécanisme de sélection multiple. Enfin, le système de visualisation flexible est un composant qui s'intègre du côté de la partie lecture en [CQRS](#) pour proposer un contenu adapté au client.

4.2.1 Éditeur 3DEvent

L'intérêt de proposer une application web se retrouve principalement dans l'accessibilité qu'elle propose. En effet, n'importe quel terminal muni d'un navigateur web peut y accéder, ce qui la rend distribuée et multi-plateforme. Les fonctionnalités graphiques proposées par WebGL sont un peu réduites par rapport à celles d'OpenGL dont l'[API](#) évolue plus vite et propose plus de flexibilité et d'optimisations. Cependant, les performances graphiques restent très correctes car le navigateur est quand même capable d'utiliser les processeurs graphiques du terminal (GPU) pour les calculs et les rendus [3D](#).

Pour faire le lien entre le modèle et l'expérimentation, l'implantation du modèle a pris la forme d'un éditeur pour la modélisation 3D haut niveau permettant de visualiser et manipuler des objets 3D de manière collaborative dans un environnement web (aussi appelée « application 3DEvent »). Au sein de la plateforme, les interactions possibles sont :

Visualiser, naviguer, utiliser les outils de transformation L'utilisateur peut, comme dans un environnement 3D classique, interagir avec la vue en utilisant la souris (survol, clic) et en bougeant la caméra (déplacements). Il peut utiliser les commandes clavier et souris pour effectuer des opérations de translation, de rotation et d'homothétie de trois manières différentes : directement dans le *viewport*, via le menu ou via la console du navigateur.

Charger des modèles 3D L'éditeur gère la plupart des formats de fichier 3D (OBJ, PLY, DAE, glTF...)

Changement de référentiel La modification des coordonnées de références (local/global) pour les différentes transformations possibles

Grid snapping Cette fonctionnalité permet d'aligner les modèles avec la grille avec un effet de magnétisme sur les intersection de la grille.

Changement de point de vue L'utilisateur peut à tout moment passer de son point de vue à celui d'un autre utilisateur. Le choix d'implanter ce type de fonctionnalité s'inscrit dans la perspective de sensibilisation de l'utilisateur au travail de ses collaborateurs. Ainsi, lors de la session, le fait de prendre le point de vue d'un collaborateur est une manière de comprendre son fonctionnement et d'imaginer ses perspectives de conception à travers l'angle de caméra qu'il aura choisi.

4.2.2 Interface utilisateur orientée tâche

Dans le but de proposer une IU proche des fonctionnalités métiers liées à la modélisation 3D, l'éditeur possède une interface orientée « tâche », en comparaison avec des IU CRUD. En effet, les IU CRUD réduisent la sémantique métier du domaine d'application à la création, la lecture, la mise à jour et la suppression, omettant toutes les subtilités que peuvent dégager ces actions en perdant l'intention de l'utilisateur dès le niveau de l'interface. Une interface orientée tâche a tendance à s'attarder sur toutes les nuances que le domaine possède en caractérisant chaque action sans subir d'effet de simplification. Cette proximité avec le métier permet de calquer directement l'interface du modèle événementiel sur l'IU et de guider l'utilisateur dans ses activités. L'utilisabilité, qualité de l'expérience utilisateur fournit par un système pour réaliser une tâche, est alors maximisée en terme

d'efficacité, d'efficience et de satisfaction. Ce type d'**Interface Homme Machine (IHM)** s'organise autour de cas d'utilisation. Cela permet, d'une part, de présenter clairement les actions (« ajouter une géométrie à la bibliothèque à partir d'un fichier » plutôt que « téléverser un fichier ») : l'intention est clairement définie. D'autre part, lorsque l'utilisateur s'apprête à faire une action, seules les informations utiles sont affichées. Enfin, l'application fournit simplement l'information dans le contexte où elle doit être présentée, évitant à l'utilisateur d'aller la chercher ailleurs. L'**IU** devient alors une couche de l'application qui nécessite d'agrégner, croiser et filtrer des données. La dénormalisation proposée par **CQRS** remédie à ce besoin dans le cadre de la consultation de données.

Présentation de l'interface

Lorsqu'un utilisateur se connecte à une scène, il a accès à une interface web (dans un navigateur) qui représente l'espace de travail collaboratif et qui lui permettant d'utiliser différentes fonctionnalités. Les deux modalités d'interaction sont le clavier et la souris. Le premier niveau de cette interface est scindée en deux panneaux :

1. L'espace **3D** consacré à la visualisation des objets et à leur manipulation dans l'environnement **3D** ;
2. La barre d'outils qui contient trois onglets :
 - (a) "Scene" contient tous les détails de la scène et des maillages qu'elle inclue ;
 - (b) "Collaboration" fournit les informations liées à la collaboration ;
 - (c) "History" liste tous les événements qui ont eu lieu dans la scène et leurs détails.

La Figure 4.1 montre quelques captures d'écran durant une session collaborative sur le modèle Rotor.

L'onglet "Scene" (Figure 4.1a) possède un bloc contenant les détails d'un maillage en cours de sélection. Cela permet d'avoir la description des propriétés de l'objet sélectionné et une manipulation de ses paramètres (position, rotation et mise à l'échelle) plus précise que via l'espace **3D** avec le cliqué / déplacé. "Scene" intègre également un espace réservé aux géométries disponibles dans la scène appelé Bibliothèque (de géométries).

L'onglet "Collaboration" (Figure 4.1b) présente la liste des collaborateurs qui participent à la scène. Chacun d'eux est décrit par son nom, son état (connecté ou déconnecté) et son rôle (administrateur, éditeur, lecteur ou autre¹). En cliquant sur un élément de la liste, l'utilisateur accède au dernier point de vue dans l'espace **3D** connu du collaborateur représenté.

1. Un rôle peut être défini par le biais du framework 3DEvent

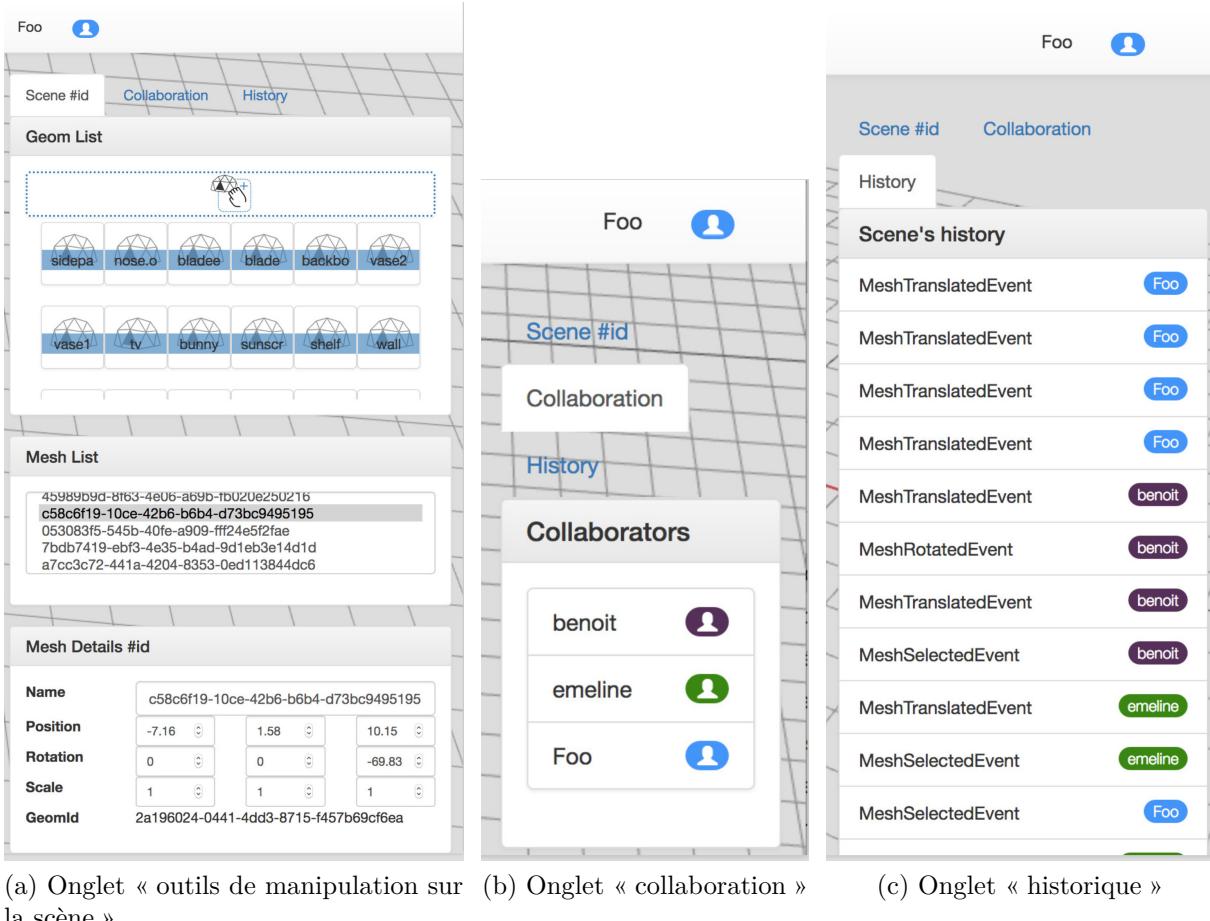


FIGURE 4.1 – Onglets du panneau latéral de l'interface

L'onglet "History" (Figure 4.1c) liste tous les événements passés dans la scène en fournissant l'accès à leur détail. Pour chaque événement, le système est capable de montrer dans l'espace 3D la différence entre l'état après l'événement cliqué $state_x$ et l'état courant $state_n$. L'utilisateur peut à partir de cette visualisation choisir de « revenir en arrière » sans perdre les données entre $state_n$ et $state_x$ car dans le système présenté ici, cela s'effectue par compensation (cf Event-Sourcing Section X).

Dans chaque onglet se trouvent différents blocs **HTML**, avec des comportements spécifiques à un agrégat et injectés dynamiquement. Ces blocs correspondent aux views de ce modèle.

Les boîtes englobantes représentent la sélection des différents collaborateurs pendant la session.

Parmi les views disponibles dans le système, une grande partie est dédiée à l'**IU** de l'application web pour le cas d'utilisation de la modélisation 3D. D'autres views sont

disponibles pour un autre type d'utilisation destinée à l'observation des comportements des utilisateurs, élément est primordiale dans une expérimentations.

Exemple d'interaction La Figure 4.2 décrit la façon dont le système traite l'exécution d'une commande de translation déclenchée par l'utilisateur et comment cette information est diffusée aux collaborateurs². Dans l'étape (a), la commande déclenchée par l'utilisateur s'adresse à l'agrégat *cube1* et contient les paramètres de la translation (vecteur x, y, z). L'agrégat, qui modélise le domaine d'un maillage, génère l'événement de translation *e1* (étape (b)) si tout est valide d'un point de vue métier. L'événement *e1* est ensuite passé à l'Event Store. Le composant responsable de la détection de conflit permet au développeur d'implémenter ses propres règles de résolution de conflit. Le composant déclenche une exception lorsque le numéro de version reçu et le numéro de version courant de l'agrégat sont identiques (Figure 4.2 étape (c)). Selon les règles métiers définies et les exceptions liées à la cohérence, l'événement peut être rejeté. Ce traitement peut être à l'origine de la génération de nouveaux événements.

Sélection fantôme

Les interactions utilisateurs doivent être adaptées à la collaboration et aux manipulations à effectuer. Pour cela, l'éditeur 3DEvent introduit la fonctionnalité de sélection « fantôme ». Lorsqu'un utilisateur souhaite sélectionner un objet de la scène, l'objet original (O_o) est cloné et devient l'objet fantôme (O_f). O_f conservent les mêmes propriétés, représenté avec de la transparence d'où le terme « fantôme ». L'objet O_f prend alors le focus de sélection pour que l'utilisateur le manipule à la place de l'objet O_o . Lorsque l'utilisateur relâche O_f , alors la modification intentée s'applique sur O_o avec le principe du *Last Write Last Win* (le dernier gagne). La Figure 4.3 représente la sélection fantôme lors de la translation du corps du rotor par l'utilisateur Foo : c'est l'objet transparent qui est manipulé alors que l'objet opaque représente sa position originale. En différenciant l'actuel objet que l'utilisateur souhaite sélectionné (O_o) de celui manipulé (O_f), l'interaction est mise en valeur sous quatre angles :

- l'ergonomie dans l'environnement 3D : O_f est un objet temporaire qui permet à l'utilisateur d'avoir une visualisation de l'objet en cours de manipulation tout en conservant le dernier état de O_o visible. O_o peut être considéré comme un point de repère visuel pour l'utilisateur lorsqu'il effectue sa manipulation. L' O_f a aussi un rôle d'intermédiaire entre l'utilisateur et la finalité de l'interaction en donnant

2. Pour que l'exemple fonctionne, la scène, la géométrie du cube et le maillage *cube1* doivent avoir été créés en amont.

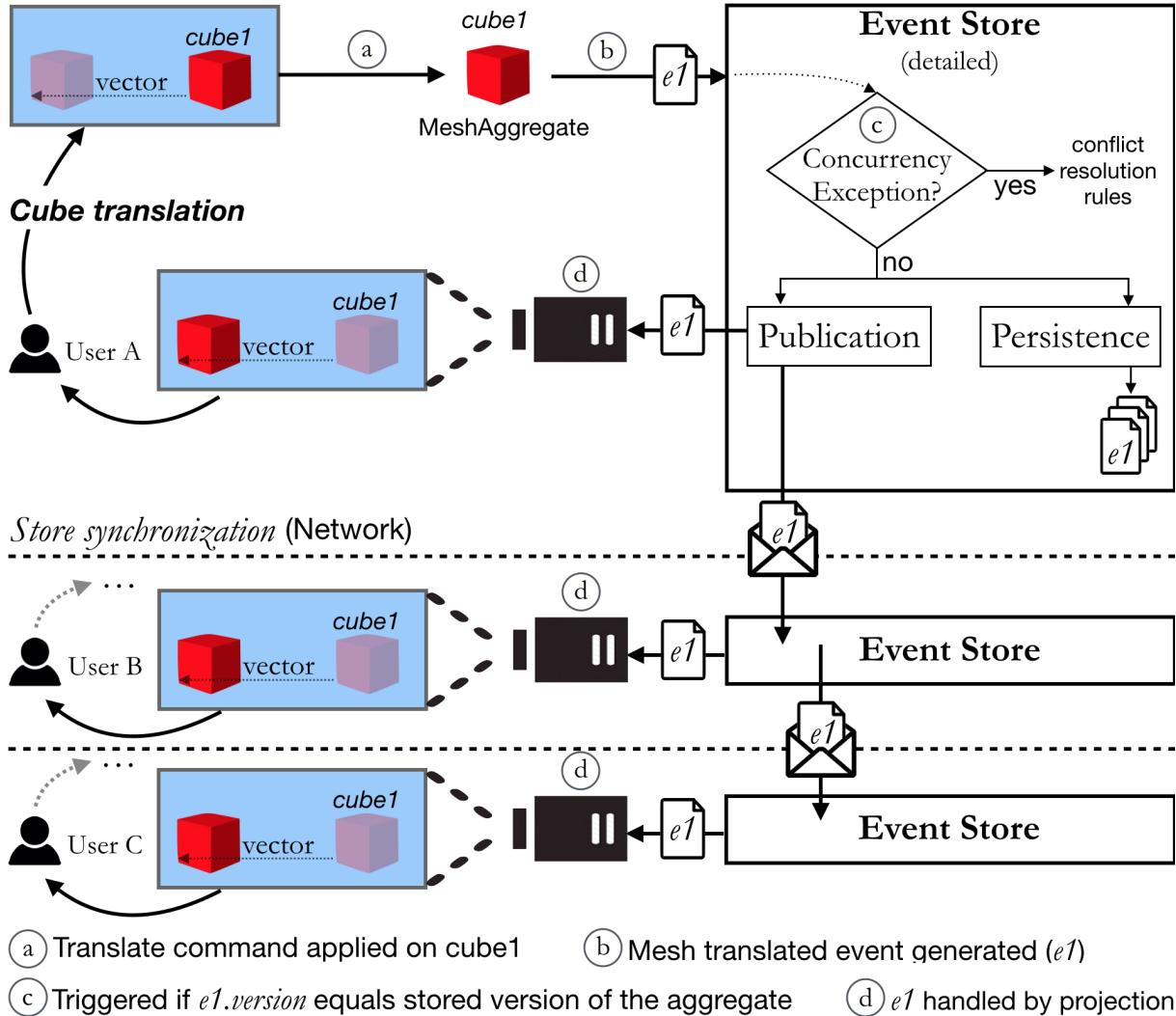


FIGURE 4.2 – Exemple d'édition collaborative où User A est connecté à User B, lui-même connecté à User C. Le cycle montre les différentes étapes du déclenchement : la commande, la génération de l'événement, la synchronisation du journal d'événements, l'impact sur le rendu des autres utilisateurs pour une translation sur un cube et le rendu visuel.

un support visuel à sa réflexion experte. Grâce à O_f , l'utilisateur peut également révoquer sa manipulation en cours sans avoir eu d'impact sur O_o en évitant des actions inutiles (faire l'action puis la défaire) pour le métier et coûteuses pour le réseau.

- la collaboration : si un collaborateur effectue une modification à destination du même O_o alors la représentation de O_o chez l'utilisateur est également modifiée. O_f par contre ne subit pas d'impact ; l'utilisateur peut continuer sa manipulation et / ou l'ajuster en fonction des nouvelles informations liées à O_o ou même révoquer sa manipulation en cours si cela lui convient.
- le métier : seules les manipulations menées à terme sont considérées comme des commandes. Cela évite d'avoir des événements qui ne sont pas pertinents pour le métier dans le journal d'événements (comme lorsque l'utilisateur change d'idée lors de l'interaction ou suite à une intervention concurrente). L'utilisateur n'a un impact sur l'application que lorsqu'une modification métier est réalisée.
- le réseau : l'information importante à faire transité est l'événement correspondant à la modification métier pas toutes les positions intermédiaires même si intuitivement l'idée de temps réel pourrait conduire à cette solution. La quantité de messages produite surchargerait à la fois le réseau et le fil d'exécution principale de l'application. En effet, [WebRTC](#) a l'inconvénient pour le moment de ne pas pouvoir s'exécuter dans un [Web Worker](#) (fil d'exécution parallèle en JavaScript). Cette solution imposerait des latences réseau et d'IU qui affecteraient gravement l'expérience utilisateur sans apporter d'informations supplémentaires à l'aspect métier de la collaboration.

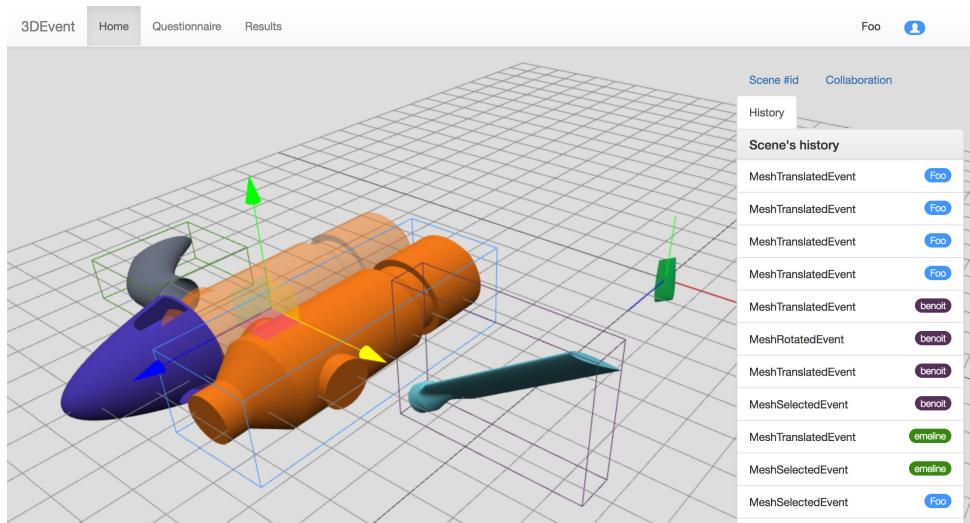


FIGURE 4.3 – Illustration de la sélection fantôme dans l'environnement 3D

4.2.3 Flexibilité de la visualisation

Dans l'approche **CQRS**, une projection est définie comme une dérivation de l'état courant à partir du flux d'événements. Pour Abdullin, « la projection est le processus de conversion (ou d'agrégation) d'un flux d'événement en une représentation structurelle. Cette dernière (qui est mise à jour au moment où le flux est parcourue) peut avoir différentes appellations : modèle de lecture persistente, vue ou état »[Abdullin, 2011]. La partie lecture du modèle (l'affichage sur interface utilisateur) bénéficie des projections en lui permettant de réduire l'afflux des événements, ne laissant filtrer que ceux qui sont pertinents pour la vue. La projection fournit une vue adaptée (filtrée, enrichie...) du flux d'événements au client. Elle peut également être utilisée pour mettre en avant des aspects experts (notifications, déclenchement d'action) ou des raisons de confidentialité. Une projection peut être créée de manière synchrone (à la volée) au fur et à mesure de la publication des événements ou de manière asynchrone et donc découpée du flux des événements.

Du fait de la nature d'un réseau **P2P**, les pairs ne reçoivent pas forcément les paquets réseau de manière ordonnée. Par conséquent, les messages peuvent arriver dans n'importe quel ordre. Qu'arriverait-il alors si un événement A (eA) nécessitant un autre événement B (eB) arrivait avant celui-ci ? Dans cette situation, le système génère une erreur en essayant d'appliquer eA sur un état inadéquat car il n'a pas d'information sur la hiérarchie d'application des événements (eB puis eA).

Pour pallier à ce problème, l'introduction du système de projection permet d'avoir un mécanisme (comme un automate fini) qui définit les transitions nécessaires pour passer d'un état à l'autre. Les transitions réalisent les actions déterminées en fonction des événements qui arrivent. Par exemple, si un utilisateur essaie d'ajouter un objet dans une scène (eA) sans avoir créé la scène (eB) la projection met en attente eA jusqu'à recevoir eB . Dans le cas où eB n'arrive jamais, la projection ne pourra jamais utiliser eA .

Projections chaque partie de l'interface est liée à une proj de la bdd les actions user et les actions des autres users passent par le même cycle pas de diff de prise en compte des evnts de l'interaction user ou de la couche reseau en CS : les actions users -> actions -> recuper info action push du servuer qui peuvent etre gerees de maniere diff

4.2.4 Bilan

L'application 3DEvent repose sur les principes et les technologies du web pour permettre de visualiser et manipuler des objets **3D** de manière collaborative en temps-réel.

4.3 Intergiciel P2P pour l'échange de données 3D pour le paradigme événementiel

L'architecture de communication décrite dans le chapitre précédent nécessite l'implémentation d'un intergiciel **P2P** compatible avec les besoins liés à la 3D, le web et la collaboration. Cette section décrit les politiques de connexion pour l'éditeur présenté dans la section précédente ainsi que les mécanismes de synchronisation nécessaire à l'échange de données dans l'application.

L'assumption est faite que tous les clients utilisent des navigateurs qui implémentent et supportent le protocole WebSocket et l'[API RTCDataChannel](#).

La topologie de l'architecture de communication repose sur la mise en relation automatique des clients par le biais du serveur pour établir une connexion [WebRTC](#). Pour ce faire, chaque client envoie son identifiant (ID) lors de sa première requête au serveur qui le stocke. Selon le paramétrage de la connectivité directe minimum établie préalablement, le serveur recherche aléatoirement l'ID d'autres clients qui satisfont la règle de connectivité. Cette règle de connectivité minimum permet d'ajuster la densité du maillage (connectivité élevée : maillage partiel dense, voire complet ; connectivité faible : maillage partiel éparse) et d'obtenir une topologie maillée adaptée aux besoins de l'application en termes de synchronisation (temps-réel ou non) ou aux capacités des appareils. Il faut noter cependant que plus la connectivité est faible, plus l'information a besoin de transiter par des intermédiaires pour atteindre tous les pairs et par conséquent le temps de transmission est augmenté (exemple d'une distribution en ligne).

Implantation du serveur

La connexion entre un pair (client) et le serveur est établie sur la base du protocole [WebSocket](#). Cette connexion bi-directionnelle est initialisée lors de la première requête du client pour récupérer le contenu de l'application. Cette connexion sert à la fois pour la phase de *signaling* lors de l'établissement d'une connexion [WebRTC](#) mais également pour que le client puisse envoyer des mises à jours originales à la base de données via les pairs reliés à la base de données.

Lors de la phase de *signaling* c'est donc la partie gestion des instances du serveur qui gère la politique de connexion des données. Dans le prototype réalisé la connectivité minimum entre les différents pairs participant à la même scène est fixée à 2. Cette politique permet de créer un maillage réseau qui n'est pas trop dense en considérant que le nombre maximum d'utilisateur dépassera pas dix.

De navigateur à navigateur

Lors de la connexion d'un nouveau client à la scène, le serveur effectue la phase de *signaling* permettant de le mettre en relation avec un autre client. Le mécanisme est répété tant que la règle de connectivité peut s'appliquer. Le client reçoit une notification de l'établissement de la connexion avec un autre client ce qui lui permet de démarrer l'échange de données.

L'API RTCDataChannel permet à chaque pair d'échanger des données arbitraires avec d'autres à partir du navigateur avec des propriétés de livraison personnalisables – fiable ou non fiable (Section 2.2.1), ordonné ou non ordonné (Section 2.2.2).

Dans 3DEvent, le choix d'avoir un transport fiable et non ordonné a été fait pour respectivement garantir l'arrivée d'une donnée émise par l'utilisateur au sein de l'application et permettre des échanges asynchrones. En cas d'arrêt soudain du serveur, si une connexion a été établie préalablement entre les clients et est toujours en cours, elle n'est pas impactée par la défaillance du serveur.

4.3.1 Données d'échange

En sachant que le modèle est conçu pour des applications web, 3DEvent a besoin d'un format de données permettant de faire communiquer des acteurs hétérogènes du système. Le format **JSON** est assez générique pour être aussi utilisé pour lors de la sérialisation et la désérialisation des événements et plus largement des paquets transmis par RTCDataChannel. L'**API** RTCDataChannel supporte beaucoup de types de données différents (chaînes de caractères, types binaires : Blob, ArrayBuffer...). Dans un environnement multi-utilisateurs tel que 3DEvent avec des données hétérogènes (3D, images, informations) l'interopérabilité est facilitée.

Le Listing 4.1 représente la structure de données utilisée sur le réseau lorsqu'un événement est transmis. Seule la partie *data* est utilisé au sein du pair pour faire les manipulations en **CQRS** car le type de l'instance d'événement manipulé est connu.

La structure représentant l'événement transporte un numéro de version « attendue » qui correspond à la version que l'agrégat doit avoir lorsque l'événement lui est appliqué. De cette manière la cohérence interne du pair est maintenu grâce à ce numéro de version. Les événements contenant les objets 3D (*importGeometryEvent*) ont une propriété contenant la géométrie sous la forme d'un Blob contenant le fichier.

```
1 {  
2   "packetId" : "567826g6-766b-f0g9676b677r",
```

```

3   "streamId" : "scene-turbine",
4   "eventType" : "MeshAddedToSceneEvent",
5   "version" : 17,
6   "data" : {
7     "sceneId": "scene-turbine",
8     "meshId": "406514c6-306b-f0f9643a037e",
9     "geometryId": "37076875-ea1c-bbd300481345",
10    "name": "blade",
11    "color": "#963912",
12    "author": "Foo"
13  }
14 }
```

Listing 4.1 – Format du Message transitant sur le réseau contenant l'événement MeshAddedToScene et ses paramètres

4.3.2 Synchronisation des données

Persistance à court terme

Le navigateur (client) offre un espace de stockage avec l'interface *Storage* de l'API Web Storage qui donne accès au `session storage` ou au `local storage`. Ce stockage est présent sur le client et fonctionne sur un système de clé / valeur. La configuration du client est également stockée localement. Grâce à ce système, il est possible d'avoir une persistance des données à travers les sessions du navigateur. Le contenu stocké correspond aux données générées par un utilisateur et par ses collaborateurs. Les répliques stockées sur chaque navigateur permettent à un utilisateur une plus grande autonomie en cas de déconnexion. C'est également un moyen de distribuer entre les clients les mises à jour qu'ils génèrent grâce au protocole de [streaming 3D](#) (cf. 4.3.2) sans passer par le serveur.

événements enregistrés sur le client. La configuration du client est également

Protocole de streaming pour la synchronisation

Il existe plusieurs méthodes de transmission de contenu au sein d'un réseau [P2P](#) que l'on peut catégoriser selon deux modes : le téléchargement (*download*) et le flux continu (*streaming*). Le téléchargement requiert que le contenu soit entièrement téléchargé pour pouvoir être lu, tandis que le flux continu peut être lu au fur et à mesure de sa récupération. Ce dernier mode, principalement utilisé pour la lecture de vidéo en ligne, s'applique bien à la transmission de contenu [3D](#) : niveaux de détails [Chu et Chan, 2012, Hu *et al.*, 2008],

4.3. Intergiciel P2P pour l'échange de données 3D pour le paradigme événementiel

progressif [Cheng *et al.*, 2009, Limper *et al.*, 2014], mise en cache [Jia *et al.*, 2014], compression / décompression [Lavoué *et al.*, 2013, Ponchio et Dellepiane, 2015, Maglo *et al.*, 2013]. Une catégorisation plus précise du flux continu peut être donnée selon quand le contenu est généré : en direct (*live* ou *push*) et à la demande (*on-demand* ou *pull*).

Le mécanisme de routage que nous avons utilisé dans [Desprat *et al.*, 2015b] est proche du routage de GNutella.

chaque client receptionne les info qui l'interesse : une scene en priorite ? les messages sont stockés localement chaque client s'occupe de maintenir sa cohérence et s'interroge les uns les autres pour cohérence globale. -> bcp de messages émis pour chaque événement sauvegardé, l'Event store publie (push) l'événement

Algorithm 4.1 Sauvegarde d'événements d'un agrégat dans l'Event Store

```
Entrée : aggregateId : string, uncommittedEvents : eventMsg[ ], version : number
expectedVersion ← version
for eventMsg in uncommittedEvents do
    newMsg ← {type : eventMsg.typeEvt, data : eventMsg, streamId : aggregateId}
    publish(aggregateId, newMsg, expectedVersion++)
end for
```

Algorithm 4.2 Ajout d'un événement dans l'Event Store

```
Entrée : streamId : string, event : EventStoreEvent, version : number
```

```
Sortie : event
```

```
stream ← streams.getOrCreate(streamId);
if stream.has(version) then
    throwExceptionVersion(stream, version)
end if
stream.data.set(version, event)
```

categories : type d'aggregat c'est un flux lié à une catégorie (exemple geometries) tous les événements liés à la catégorie.

metadata phase sync permet à un nouveau nœud de récupérer toutes les informations manquantes et de les demander de manière répartie à tous les nœuds

L'algorithme 4.3 décrit la synchronisation d'un nœud avec un autre nœud dans l'Event Store partagé. Le nœud courant demande le différentiel (*diff*) entre ses streams et ceux du nœud *node* pour connaître quels sont les streams à synchroniser. Le plus rapide répond et ça continue jusqu'à synchronisation. ignore si déjà présent.

Gestion des événements des agrégats

Une connexion RTCDataChannel peut être configurée selon les critères (Section) de

ref section
config

Algorithm 4.3 Synchronisation d'un noeud de l'Event Store partagé

```

Entrée : node : ClusterNode
Sortie : nodeState == synchronized
nodeMetadata <- node.getMetadata()
if nodeMetadata! = {} then
    streamsToSync <- metadata.getDiff(nodeMetadata)
    while streamsToSync > 0 do
        streamToSync <- streamsToSync.pop()
        events <- node.getEvents(streamToSync)
        if !streamToSync.has(version) then
            for event in events do
                processEvent(event.streamName, event, event.version)
            end for
        end if
        streamsToSync <- metadata.getDiff(nodeMetadata)
    end while
end if
node.endSync()

```

fiabilité et d'ordonnancement. Dans l'intericiel de 3DEvent, cette configuration est : non fiable et non ordonnée. C'est l'application qui est en charge de « ré-ordonner » les événements. En effet, les événements sont associés à des agrégats. Comme chaque agrégat possède un flux d'événements dédié et numéroté il est alors simple de les ordonner lorsque. Lors de la synchronisation (Section), les événements manquants sont demandés successivement à tous les pairs. Lors qu'un pair reçoit les événements il les stocke dans le flux correspondant à l'agrégat. Si un événement est manquant, les suivants sont quand même stockés en laissant l'espace de l'événement manquant dans le tableau. L'événement est redemandé jusqu'à son obtention, laquelle provoque le déblocage de l'état de l'agrégat jusqu'au prochain événement manquant (ou la fin du flux). Les événements qui ont été stockés « en attendant » permettent à l'application d'être directement en capacité de poursuivre la construction de l'état de l'agrégat. Cette mécanique tire avantage des Snapshots (présentés dans) qui réduisent la taille des flux en mémoire pour chaque agrégat qui part déjà d'un état avancé.

ref section
sync

ref section
sync

4.4 Résumé des choix techniques

TypeScript et JavaScript JavaScript est un langage orienté objet à prototype principalement utilisé dans les pages web interactives (scripts) mais aussi sur les serveurs (ex : Node.js). L'application est implémentée en TypeScript. TypeScript est un langage libre et

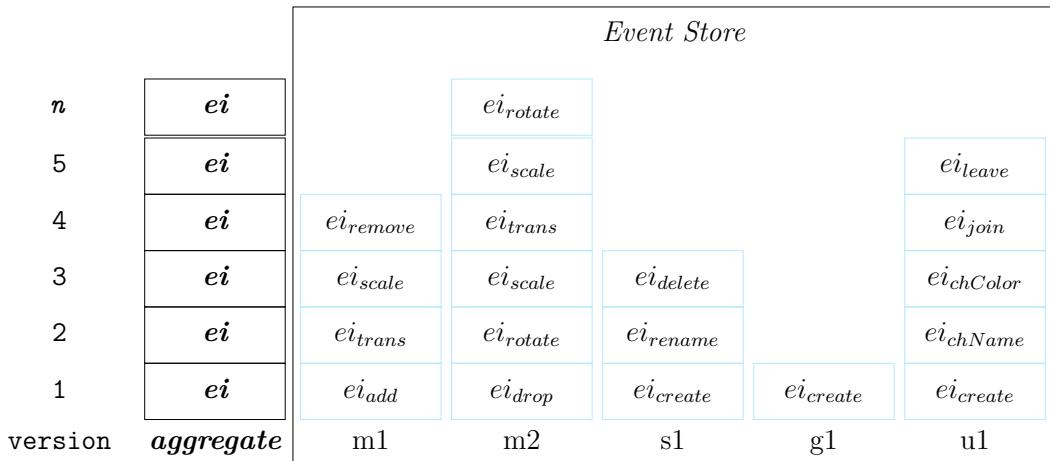


FIGURE 4.4 – Exemple d’agrégats

Structure d’un agrégat et ses versions. Chaque version est un état de l’agrégat qui correspond à l’empilement des instances d’événements (ei) qu’il contient. Les types des événements sont relatifs au type d’agrégat dans lequel il est contenu.

open-source développé par Microsoft. C’est un sur-ensemble de [JavaScript \(JS\)](#), i.e. il peut intégrer du [JS](#)). Le code écrit en TypeScript est transpilé en [JS](#) pour être interprété par le moteur [JS](#) d’un navigateur. TypeScript intègre la notion de type qui permet d’améliorer et sécuriser la production de code [JS](#).

Node.js Node.js est un environnement d’exécution [JS](#) open-source et cross-plateforme qui permet d’executer du code [JS](#) côté serveur.

Rendu 3D The 3D visualisation is done using the Three.JS library

WebRTC Even though WebRTC API is native in some web browsers, the cross browser compatibility is not well handled. Therefore, in order to ease specific development such as data processing, we used the simple-peer library⁵ which provides an abstraction level over the web browser API guaranteeing cross browser compatibility. The server instances are based on Node.JS and provide HTML and CSS contents. The WebRTC API on Node.JS is enabled through the node-webrtc module¹.

Base de données NoSQL L’évolution de l’utilisation du web en tant que plateforme applicative a encouragé le changement dans le stockage des données pour de nouveaux besoins supportant de larges volumes de données (comme les données 3D). Une base de données [Not Only SQL \(NoSQL\)](#) fournit un schéma libre et dynamique ainsi qu’une API de requête riche pour la manipulation de données. De plus, la possibilité d’enrichir

un document « à la volée » facilite l'évolution des objets (3D) et la maintenance. de l'application. 3DEvent intègre un système de persistance sur le long terme caractérisé par une base de données [NoSQL](#).

La base de données ([NoSQL](#)) conserve tous les événements qui se sont produits dans une scène. La mise en place d'une base de données centralisée apporte de la robustesse au système en lui fournissant un référent sans toutefois le surcharger ainsi qu'une expérience utilisateur transparente limitant les interruptions de service.

Base de données fonctionnelle pour l'event store The long term persistence relies on Event- Store® (Figure 3d), an "open-source, functional database with Complex Event Processing in JavaScript"⁶. It stores data as a series of immutable events over time, according to event-sourced application requirements. It offers a continuous projection service reacting to events as they are written. It is accessed from server instances via HTTP protocol. Each exchange of data is serialised in JSON.

4.5 Bilan

Chapitre 5

Expérimentations

Sommaire

5.1	Cas d'étude : Assemblage collaboratif d'objets 3D dans un environnement web	99
5.2	Expérimentation 1 : preuve de faisabilité	100
5.2.1	Présentation de l'expérimentation 1	100
5.2.2	Résultats et discussion	102
5.2.3	Conclusion de l'expérimentation 1	103
5.3	Expérimentation 2 : intégration du framework événementiel	104
5.3.1	Présentation de l'expérimentation 2	104
5.3.2	Résultats et discussion	106
5.3.3	Conclusion de l'expérimentation 2	108
5.4	Comparaison entre l'expérimentation 1 et l'expérimentation 2	109

5.1 Cas d'étude : Assemblage collaboratif d'objets 3D dans un environnement web

Comme nous l'avons vu, dans le cadre de la modélisation 3D collaborative, il existe plusieurs types d'application (**CAO** collaborative, **BIM** ...). Ces domaines d'activité ont un point commun : les fonctionnalités de modélisation 3D récurrentes sont celles permettant l'assemblage d'objets 3D de manière collaborative. Dans la mise en situation, les utilisateurs ont déjà une modélisation précise de leur modèle réalisée par exemple à l'aide de logiciels dédiés. L'utilisateur est responsable de fournir un modèle compatible avec le prototype, notamment concernant le format et la taille du fichier et la sécurité du modèle qu'il partage (tatouage et droits d'accès). Le cas d'étude présente la mise en commun de ces objets pré-conçus dans un environnement web.

L'**expérimentation 1** concerne les travaux réalisés dans [Desprat *et al.*, 2015b] et [Desprat *et al.*, 2015a]. J'utilise alors le prototype décrit dans la Section et je démontre ref section la faisabilité concernant l'architecture de communication décrite dans la section 3.3.2.

L'**expérimentation 2** concerne les travaux réalisés dans [Desprat *et al.*, 2016] et [Desprat *et al.*, 2017]. J'utilise le prototype décrit dans la Section et je démontre l'intérêt ref section auprès des utilisateurs d'une **EDA** pour la visualisation et la manipulation d'objets 3D reposant sur le modèle événementiel présenté dans la section 3.2 ainsi que l'architecture de communication décrite dans la section 3.3.4.

Globalement, les objectifs des travaux de cette thèse soulignent plusieurs aspects relatifs à la fiabilité du modèle et à son implantation de deux manières :

- fonctionnelle : manipulation et visualisation 3D, historique ;
- et technique : récupération de l'information, cohérence des données, disponibilité du réseau.

Les deux expérimentations présentées ont en commun de proposer une enquête utilisateur sur l'usage de leur prototype respectif. Le cas d'étude regroupe les résultats des enquêtes utilisateurs menées sur la praticité de l'interface, la réactivité du système et la cohérence des scènes manipulées lors des essais de collaboration. Il est également fait référence à l'impact de l'architecture de communication sur l'expérience utilisateur : est-ce transparent pour l'utilisateur ? adapté ? plus efficace que l'architecture de l'expérimentation 1 ?

5.2 Expérimentation 1 : preuve de faisabilité

Les travaux de Desprat et al. [Desprat et al., 2015b, Desprat et al., 2015a] présentent une preuve de faisabilité concernant l'architecture de communication dans une application d'assemblage d'objets 3D : qu'apporte une architecture hybride à la collaboration temps réel ? Est-il possible techniquement de réaliser une architecture pour l'échange de ressources 3D de manière fiable et efficace uniquement avec des technologies web ? Quels sont les apports pour la collaboration, les utilisateurs et la réalisation de la tâche ? Un des objectifs de cette expérimentation est de démontrer la faisabilité de notre approche réseau hybride, avec une attention particulière à l'égard de l'expérience utilisateur.

5.2.1 Présentation de l'expérimentation 1

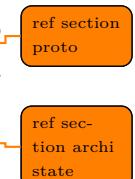
L'expérimentation propose de répliquer une collaboration réaliste entre différents participants travaillant à distance sur des tâches d'assemblage de modèles 3D. L'expérimentation reprend le cas d'étude de l'assemblage d'objets 3D (Section 5.1).

L'expérimentation est basée sur un prototype d'application développé selon l'implantation présentée dans la Section . Le prototype est développé sur une plateforme web et consiste en un application de modélisation 3D collaborative multi-utilisateurs pour démontrer la faisabilité de l'architecture présentée dans .

Tâches à effectuer. L'expérimentation consiste en quatre épreuves qui partagent la même procédure. Les modèles utilisés dans chaque épreuve sont décrits dans le tableau A.4 : *Wind Turbine*, *Pick up* et *Castle*.

Les épreuves sur les modèles *Wind Turbine* et *Pick Up* ont le même objectif : assembler un modèle dont les utilisateurs possèdent l'image, composé de différentes pièces qui sont téléchargées par les utilisateurs. L'image correspond à l'assemblage final à obtenir et indique les différentes parties qui le composent (type et quantité). Ces deux épreuves s'arrêtent lorsque les participants le notifient oralement, ou durent dix minutes maximum chacune.

Le modèle *Castle* est utilisé dans deux épreuves : *Castle from server* et *Castle from peer*. L'objectif de ces épreuves diffère un peu des épreuves précédentes car l'épreuve s'effectue sur un modèle de château en kit (composé d'une dizaine d'objets différents : tour, murs, escalier...) : les participants ont dix minutes pour construire un château de manière collaborative en faisant appel à leur créativité. Dans l'épreuve *Castle from server*, les objets sont récupérés automatiquement à partir du serveur ; dans l'épreuve *Castle from peer*, les pairs peuvent ajouter de nouveaux objets du kit qu'ils possèdent sur leur machine.



En utilisant les outils de l'éditeur, les informations liées aux collaborateurs (représentation dans l'environnement **3D** de leur position ou de leur sélection) permettent à un utilisateur de manipuler les pièces (sélection, rotation, translation, homothétie) de manière collaborative pour contribuer à atteindre l'objectif.

Population L'expérience a été conduite sur trois groupes de deux ou quatre participants chacun. Une épreuve sur le modèle *Wind Turbine* a été effectuée avec six participants en simultané. Durant l'expérimentation, les utilisateurs étaient sur le même réseau que le serveur (**LAN**). Les participants étaient des étudiants de Master ou de Doctorat en Informatique plutôt familiers avec les environnements **3D**. Les participants étaient autorisés à communiquer entre eux (oralement, par chat...).

Procédure L'expérimentation consiste en quatre épreuves qui partagent la même procédure :

1. Phase d'essai
 - (a) explication du contexte et de l'expérimentation
 - (b) prise en main du système, familiarisation avec l'interface
2. Phase de collaboration : réalisation de l'épreuve (se répète autant de fois que d'épreuves)
 - (a) Présentation de l'épreuve et de son objectif
 - (b) Initialisation : nettoyage de la scène et chargement des modèles
 - (c) Réalisation de l'objectif
3. Phase de retour d'expérience : remplissage questionnaire et discussion informelle sur les épreuves.

Initialisation L'épreuve est décrite et l'objectif de l'épreuve est présenté à tous les participants. Pour chaque phase de collaboration, l'application nettoie les données issues de l'épreuve précédente. À l'exception des pièces du modèle *Castle* qui sont chargées en totalité sur le serveur dans *Castle from server* et les pairs *Castle from peer*), les pièces du modèle de l'épreuve sont distribuées de manière aléatoire entre les différents participants.

Données collectées Entre chaque épreuve, les participants ont exprimé certains retours qualitatifs dont il a été pris note. Le type d'interaction entre les participants est observé lors des épreuves.

Questionnaire Au début de l’expérimentation, les participants reçoivent et prennent connaissance du questionnaire (voir Annexe A.3.2). Ce questionnaire est rempli à plusieurs moments au cours de l’expérimentation. Au début, le participant décrit sa situation, sa familiarité avec les environnements 3D, le type de machine et de navigateur qu’il utilise. À la fin de chaque épreuve il répond aux questions concernant l’épreuve qu’il vient de passer. À la fin de l’expérimentation, il répond aux questions d’ordre plus général sur son expérience et les améliorations envisageables.

5.2.2 Résultats et discussion

Les participants ont globalement été satisfaits des résultats issus de la collaboration ainsi que du rendu visuel des assemblages effectués durant les épreuves. Cela est également dû au fait qu’ils ont atteint les objectifs à chaque fois (dans le temps imparti) : réalisation de l’assemblage du modèle 3D présenté sans frustration. Ils se sont également « amusés » sur les épreuves avec le *Castle* car ils étaient libres dans la création et souhaitaient continuer l’épreuve au-delà du temps imparti. L’utilisation de canaux de communication externes à l’application a été reportée plusieurs fois, principalement sous formes d’échanges oraux. Ces échanges concernaient l’état de leur application et ce qu’ils étaient en train de faire ou ce qu’ils souhaitaient faire. Cette interaction provient, d’après les participants, du manque de retours visuels différenciés sur les manipulations effectués par les collaborateurs (exemple : pas de couleur différente pour la sélection d’objet par un autre utilisateur). Cela peut plus généralement s’interpréter comme un manque sensibilisation à l’environnement collaboratif. L’IU a été bien appréciée, parfois jugée « trop simple » par certains utilisateurs. Ce choix avait été fait pour faciliter l’usage, l’apprentissage et l’adoption de l’IU et s’est révélé anecdotique lors de l’expérimentation car le panel de personne de participants était familier de ce genre d’environnement. Les fonctionnalités liées à la manipulation d’objets reçoivent une bonne évaluation, à l’exception de l’importation de modèles. Cela est dû, dans ce prototype, au fait que l’application ne peut traiter et transmettre des modèles trop lourds. Un participant a vu sa fenêtre « geler » (navigateur Chrome) lors de l’épreuve *Castle from peer*. Cela l’a mené à quitter la session et à revenir sur la scène pour pouvoir continuer de participer à la session collaborative. Sans perturber la session en cours pour ses collaborateurs, le participant a pu revenir sur l’application, reprenant la collaboration avec les autres participants (rétablissement des connexions après un crash). En cela le participant a apprécié la robustesse de l’application, sans être perturbé très longtemps par l’interruption. Concernant la fluidité des manipulations et de la visualisation au sein de l’application durant la collaboration, les participants

n'ont pas ressenti de latence excessive (au-delà de 10s). Ils ont qualifié l'application de « temps réel » plutôt que d'« interactive ». La variation du nombre d'utilisateurs sur les différentes épreuves n'a pas altéré la qualité du rendu et de réseau pour le participant.

5.2.3 Conclusion de l'expérimentation 1

Cette expérimentation présente différentes épreuves sur le cas d'étude lié à cette thèse. Elle repose sur un modèle utilisant une architecture de communication hybride combinant client-serveur et P2P. Le client est responsable de proposer un environnement 3D intégré à l'interface, permettant la manipulation et la visualisation des objets 3D manipulés de manière collaborative. Afin de pouvoir collaborer, il héberge également les connexions nécessaires pour communiquer ses mises à jour vers les autres pairs (autant de connexion WebRTC DataChannel que de pairs) et vers la base de données via le serveur (une seule connexion WebSocket). Les modifications sont transmises sous forme de différentiel d'état et sont stockées sur les pairs localement et sur la base de données de manière distante. Le réseau P2P est composé uniquement de client producteurs de données reliés de manière complète. Les évaluations liées à l'expérimentation sont plutôt encourageantes, même si certains points doivent être améliorés. Concernant la fonctionnalité d'import de modèle, la transmission par les pairs du modèle a causé des latences sur les pairs destinataires (gel de la fenêtre). Afin de réduire la latence sur des scènes aussi voire plus grandes, deux alternatives sont à étudier : l'utilisation d'un rendu progressif et compressé ainsi qu'une meilleure répartition de la charge entre les pairs pour la distribution des données. Dans ce dernier cas, l'utilisation d'un maillage partiel et d'une distribution des modèles incluant les pairs permettrait de responsabiliser davantage les pairs en désengorgeant un peu le serveur et surtout le réseau P2P. Concernant l'amélioration des fonctionnalités liés à l'interface, l'ajout de retours visuels liés aux interactions collaboratives ainsi que la visualisation de l'historique sont indispensables. Une évaluation quantitative approfondie pourrait permettre de mieux examiner les apports de l'architecture de communication hybride par rapport aux architectures classiques dans le contexte de la modélisation 3D collaborative. Cela pourrait se concrétiser par l'examen de la collaboration en récupérant le journal des actions, l'impact sur la visualisation (Frame Per Second), et l'observation des échanges WebRTC en entrées / sorties.

5.3 Expérimentation 2 : intégration du framework événementiel

Le cas d'étude présenté dans la section 5.1 est de nouveau utilisé dans le cadre de l'expérimentation 2 : plusieurs participants sont réunis dans l'environnement collaboratif pour assembler différentes parties d'un modèle. Ces travaux sont réalisés sur la base des contributions liées aux EDA présentées dans cette thèse : [quels sont les apports d'un EDA](#) pour les utilisateurs ? quelles informations pouvons nous extraire de la collaboration grâce aux événements ? est-ce que l'architecture de communication est adaptée aux échanges (format, fréquence, résilience, robustesse) ?

ref section
event

5.3.1 Présentation de l'expérimentation 2

Cette expérimentation a été réalisée sur la base des travaux de Desprat et al. [Desprat et al., 2016, Desprat et al., 2017]. Dans ces contributions, je souligne la fiabilité du modèle et de son implantation de deux manières :

- fonctionnelle : manipulation et visualisation 3D, historique ;
- et technique : récupération de l'information, cohérence des données, disponibilité du réseau.

Quant à l'observation du comportement des participants, elle est effectuée de manière quantitative (outils de surveillance) et de manière qualitative (questionnaire) lors de l'exécution d'une tâche coopérative au sein de l'application.

ref ques-tionnaire
annexe

Tâche à effectuer Les participants devaient assembler les différentes parties d'un modèle 3D en utilisant l'application 3DEvent et les fonctionnalités décrites précédemment pour que le résultat corresponde à l'assemblage donné en exemple (images). La complexité de la tâche est modulée selon deux facteurs : le nombre de parties composant le modèle et le nombre de collaborateurs. Afin de permettre un apprentissage et une utilisation facile de la manipulation des objets 3D il a été choisi de conserver des manipulations de haut niveau.

Population L'expérience a été conduite sur 12 groupes de deux ou trois participants chacun. Chaque participant était localisé en France dans une zone urbaine avec de bonnes infrastructures, en travaillant sur des réseaux distincts avec une bonne connexion internet (au moins 20Mb/s) pour éviter d'avoir des latences extrêmes (>10 secondes) au cours des expérimentations. Les participants étaient des étudiants de Master ou de Doctorat

en Informatique (pas nécessairement familiers avec les environnements 3D). Ils étaient autorisés à communiquer entre eux.

Procédure Les modèles utilisés lors de l'expérience sont décrits dans le Tableau A.5. L'expérimentation se déroule en trois phases :

1. Phase d'essai : Chaque participant s'entraîne pendant 5-10 minutes sur un modèle de test dans l'application pour se familiariser avec l'interface et les fonctionnalités proposées.
2. Phase solo : Le participant effectue un assemblage du modèle Rotor ou Camera Box.
3. Phase de collaboration : Un groupe de participants réalise deux assemblages sur (i) un petit modèle (10 ou 12 parties) puis (ii) un plus gros modèle (16 parties). La phase de collaboration a été répétée six fois : trois fois avec un groupe de deux participants et trois fois avec un groupe de trois participants.

Comme les participants pouvaient participer à différentes configurations de groupe durant la phase de collaboration, plusieurs modèles 3D avec des caractéristiques similaires (nombre de parties et nombre de triangles) ont été présentés pour éviter les biais liés à l'apprentissage.

Initialisation Pour chaque phase, l'application a été initialisée par le chargement des parties du modèle dans la bibliothèque d'objets chez chaque participant. Les parties des objets ont volontairement été positionnées (rotation et homothétie) aléatoirement afin que les utilisateurs aient à manipuler les différentes fonctionnalités. Cette configuration nous a permis d'observer l'activité à l'intérieur de chaque groupe durant l'expérimentation. Chaque participant a reçu une image de l'assemblage à réaliser (la tâche à compléter).

Données collectées Pour chaque expérimentation et chaque participant, le temps de complètement, le nombre d'événements générés et l'horodatage de chaque événement pour observer le complètement de la tâche en terme de vitesse et d'efficacité ainsi que les effets sur le temps d'implication d'un collaborateur selon le nombre de collaborateurs.

L'enregistrement des données commence lorsque le premier événement sur la scène initialisée est généré (horodatage du premier événement) ; et il s'arrête lorsque le groupe indique que la tâche à compléter est terminée (horodatage du dernier événement).

Questionnaire En dehors des données collectées, les participants devaient remplir un questionnaire basé sur l'expérimentation pour exprimer leurs retours qualitatifs concernant leur expérience (Annexe A.4.2).

Le questionnaire est inspiré de [Lewis, 1995], qui permet d'évaluer la facilité d'utilisation du système et l'implication dans la collaboration de chacun des participants. Une échelle de notation sur sept points a été utilisée, (1 : pas d'accord ; 7 : d'accord) pour avoir assez de points de discrimination [Lewis, 1993]. Au cours des différentes sessions collaboratives, l'application a produit plusieurs centaines d'événements (environ 300 par session). Les expérimentations ont été réalisées sur plusieurs types de dispositifs dont un *smartphone* avec une connexion 4G.

5.3.2 Résultats et discussion

La Figure 5.1 montre quelques captures d'écran durant une session collaborative sur le modèle *Rotor* ; et la Figure 5.2 montre les premiers événements enregistrés dans la base de données. Les boîtes englobantes représentent la sélection des différents collaborateurs pendant la session.

mettre
ailleurs ?

Analyse des interactions Les traces des utilisateurs récupérées au cours des expérimentations sont la base du travail d'analyse présenté ici. Ces traces, composées d'événements générés par les utilisateurs, permettent de savoir qui (Figure 5.3a) a fait quoi (Figure 5.3b) lors des sessions collaboratives. Généralement, le « qui » est assez facile à retrouver lors de la récupération des traces. Le « quoi » en revanche nécessite que les notifications aient une signification précise et proche du métier. Grâce au travail de découpage et de dénomination des événements effectué en amont, le journal d'événements (*log*) indique précisément tout ce qui s'est passé lors de la session du point de vue du métier. Cette fonctionnalité est intéressante dans le contexte de la traçabilité des données et lors d'audits sur l'assemblage.

Figure 5.3 montre deux angles d'enregistrement d'une session sur le modèle *living room*. Au début de la session, beaucoup d'objets sont ajoutés (*meshAddedToSceneEvent*). Seul un utilisateur a ajouté un objet en utilisant la fonctionnalité pour déposer un objet à un endroit spécifique de la scène (*meshDropped*). Le nombre d'événements concernant la sélection et la désélection est à peu près similaire aux exceptions. La différence s'explique par le fait que l'événement de désélection n'est pas déclenché lors d'un changement d'objet sélectionné, car la désélection n'est pas effectuée explicitement par l'utilisateur. Dans les premiers moments, les participants ont fortement interagi (jusqu'à 20 événements en 15 secondes). Cela s'explique par le recours massif à l'ajout d'objets dans la scène pour composer le modèle et la mise à l'échelle de ceux-ci (du au positionnement arbitraire des objets de la bibliothèque). Ensuite, les trois utilisateurs interagissent ensemble durant quelques

minutes avant que l'un d'entre eux ne quitte et ne reviennent quelques secondes plus tard (informations récupérées à partir du journal d'événements lié aux agrégats utilisateurs). Enfin, la diminution du nombre d'événements montre la fin de l'activité, les utilisateurs achevant la tâche et ajustant les derniers objets.

L'implication d'un utilisateur peut être perçue par le prisme de la fréquence de ses contributions et la variété de fonctionnalités utilisées, c.à.d. les différents types d'événements produits. L'analyse d'une session apporte plusieurs indicateurs tout au long de la session. Par exemple, l'absence d'un utilisateur pendant une longue période peut indiquer une déconnexion. Ou encore, l'utilisation trop fréquente d'un type d'événements (ou d'un motif d'événements répété) peut montrer une faiblesse de l'ergonomie de l'interface. Ce dernier aspect est illustré par un exemple dans la Figure 5.3b à 45s, où un nombre élevé de MeshScaledEvent est détecté. A posteriori, je me suis rendu compte que la fonctionnalité n'était pas suffisamment calibrée pour l'échelle du modèle et nécessitait plusieurs manipulations pour réaliser la bonne transformation.

Questionnaires Après chaque expérimentation, le participant a rempli directement un questionnaire à propos des phases solo et collaboratives qu'il a effectuées via le formulaire en ligne. Les résultats obtenus sont compilés dans la Figure 5.4 sous la forme de boîte à moustaches. Cette représentation est un moyen rapide de figurer le profil essentiel des résultats des mesures quantitatives effectuées. Globalement, les tâches ont été réalisées plus rapidement et plus efficacement de manière collaborative que de manière solo. La facilité d'utilisation et la simplicité de l'interface sont également soulignées positivement par plusieurs utilisateurs. Parmi les retours d'expérience négatifs, la stabilité du réseau a parfois amené un peu de frustration chez certains participants. Cependant, les participants ont trouvé que la cohérence de l'environnement lors de la collaboration et la récupération des données était plus qu'acceptable. Cette remarque s'accompagne également du fait que la distribution des données s'est effectuée correctement, leur permettant de coopérer efficacement.

Durant l'une des expérimentations en phase collaborative, un participant a affiché à certains moments une latence de plus de 10s ; mais le groupe nous a notifié que cela n'avait pas affecté la collaboration. Pour l'ensemble des expérimentations, quelques conflits ont été levés sur différentes opérations et à différents niveaux. Au niveau réseau (mauvaise version, désynchronisation), la politique en place est d'annuler l'événement et de resynchroniser les utilisateurs entre eux. Au niveau des utilisateurs (opérations opposées sur le même objet), la résolution du conflit doit passer par un canal externe (chat) pour que les utilisateurs se mettent d'accord.

Dans toutes les expérimentations menées, le but a été atteint dans un même ordre de temps (10-12 minutes). La facilité d'utilisation du système est bien notée, mais il reste encore quelques aspects à améliorer. Pour modérer ces résultats, il est important de rappeler que bien que les modèles ne sont pas très complexes et que tous les participants étaient débutants sur le système et n'étaient pas forcément familiers avec ce genre d'application. Le fait que l'application repose sur des technologies web a également joué en faveur de l'appropriation de l'application, car il a semblé assez naturel aux participants de se rendre à l'adresse internet donnée (sans rien installer) pour effectuer les tâches en manipulant un média (3D) inhabituel pour ce genre de plateforme. De plus, on peut également supposer que le prototype créé pour l'expérimentation correspond bien à l'objectif d'assemblage coopératif d'objets 3D puisque les tâches ont été réalisées rapidement. Sur une échelle de « non-interactif » à « temps réel » les participants ont qualifié l'application de « quasi temps réel ».

La satisfaction générale à propos de l'expérimentation et la satisfaction concernant la collaboration et l'expérience utilisateur montrent que les participants ont apprécié positivement de faire de la modélisation collaborative 3D dans un navigateur web. Quant à savoir si le nombre d'utilisateurs améliore à la fois l'efficacité et la rapidité du complètement de la tâche, les participants ont généralement été d'accord.

5.3.3 Conclusion de l'expérimentation 2

L'expérimentation 2 s'intéresse au cas d'étude présenté dans la section 5.1, extraite de [Desprat *et al.*, 2017], et utilise un prototype issu de [Desprat *et al.*, 2016] (voir Section) reposant sur un modèle original orienté événements pour la modélisation 3D collaborative.
ref proto

En menant plusieurs épreuves au sein de la collaboration sur différents modèles, je propose un large panel de situations liées aux cas d'études permettant d'évaluer les capacités de coordination et de coopération dans la réalisation des tâches.

L'expérimentation, construite sous l'angle de l'expérience utilisateur, propose ainsi d'évaluer l'utilisabilité de l'interface et les effets de l'architecture réseau utilisée. Les utilisateurs ont globalement exprimé leur satisfaction sur leur expérience générale : objectifs atteints, interface facile à prendre en main, collaboration facilitant la réalisation de la tâche. Concernant l'aspect réseau pour la collaboration, ils ont souligné la bonne récupération du système dans les cas de désynchronisation. Ces derniers n'apparaissent que lors d'une sélection multi-utilisateurs du même agrégat et sont donc plutôt rares. De plus, les utilisateurs étant sensibilisés au fait qu'un autre utilisateur a déjà sélectionné l'objet, ils sont ainsi prévenus d'éventuels conflits.

5.4 Comparaison entre l'expérimentation 1 et l'expérimentation 2

Les deux expérimentations présentées concernent le même cas d'étude mais reposent sur deux prototypes conçus sur la base de modèles utilisant des paradigmes différents.

L'expérimentation 1 repose sur un modèle orienté états et présente principalement une preuve de concept et de faisabilité concernant l'architecture de communication hybride. Le passage à l'échelle est limité car le réseau est un maillage complet, ce qui sollicite beaucoup les pairs. Le serveur, lié à la base de données, reçoit également beaucoup de requêtes, tant en écriture qu'en lecture. Le mécanisme de verrouillage évite les conflits de sélection multi-utilisateurs, mais restreint l'accès aux objets de la scène aux collaborateurs.

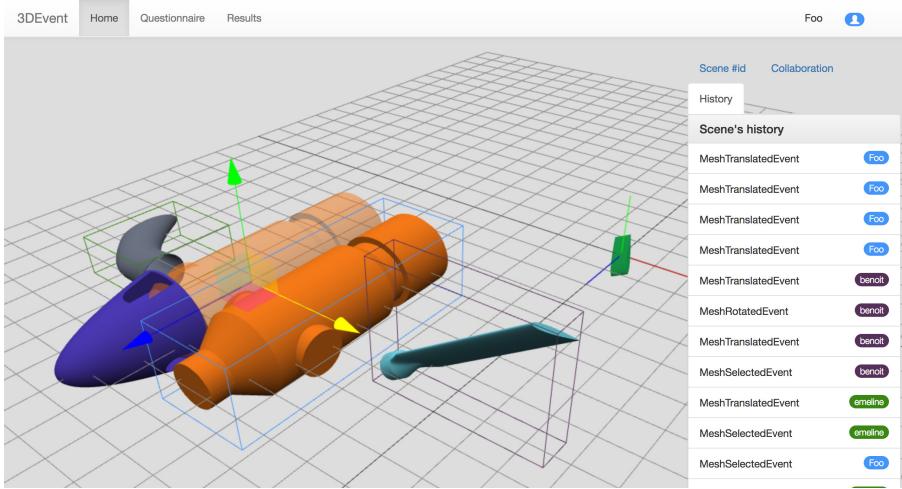
L'expérimentation 2 repose sur un modèle orienté événements et présente deux aspects des contributions développées dans cette thèse. Les événements produits permettent de décrire finement ce qui se passe durant la collaboration grâce à l'implantation des principes du DDD. L'analyse est possible grâce aux outils dérivés du CQRS. En cela, beaucoup d'aspects liés à l'observation de l'expérience utilisateur lors de la collaboration sont implicitement présents dans le prototype. La gestion des événements au sein du réseau est effectuée dans d'un journal d'événements partagé, dont la cohérence repose sur l'ordre causal des événements d'un agrégat.

Entre l'expérimentation 1 et l'expérimentation 2, le changement de paradigme a apporté aux différents utilisateurs de nouvelles fonctionnalités permettant d'améliorer leur expérience de collaboration, notamment du point de vue de la gestion des données métier. Par exemple, la validation systématique de leurs commandes à la base de chaque action dans l'expérimentation 2 évite de générer des événements problématiques. Cela est intensifié par l'utilisation de technologies comme JavaScript dans un navigateur web où il est très facile d'injecter du code permettant d'effectuer des actions sans passer par l'IU. Notamment lorsque l'utilisateur passe par l'IU, les erreurs de saisies sont plus fréquentes. D'autre part, la gestion de la cohérence utilisée dans l'expérimentation 2 est beaucoup plus permissive que celle de l'expérimentation 1. Les utilisateurs ont de ce fait exprimé moins de frustration. Enfin, dans l'expérimentation 2, les participants ont produit de plus grosses quantités de données sans ressentir de frustration, de latence excessive ou de problème de gel de fenêtre. L'expérimentation 2 ajoute des mécanismes de sensibilisation à la présence des autres utilisateurs par rapport à l'expérimentation 1 qui participe à l'évitement de conflit. L'observation des participants indique qu'ils ont moins tenté de travailler sur des objets déjà en cours d'utilisation lorsque les mécanismes sont activés,

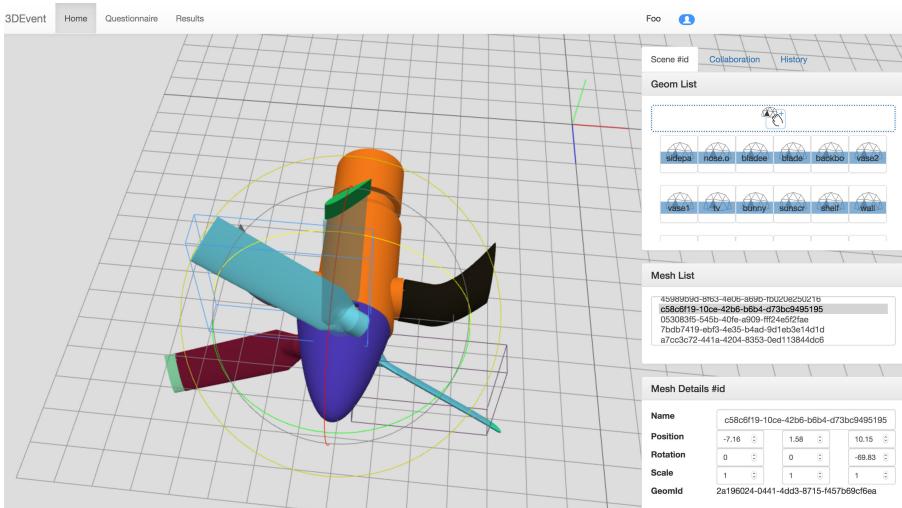
5.4. Comparaison entre l'expérimentation 1 et l'expérimentation 2

et ils ont donc généré moins de conflits. Pour compenser l'autorité d'un utilisateur sur un objet par le mécanisme de verrouillage dans l'expérimentation 1, les utilisateurs se sont appuyés sur le mécanisme de sélection fantôme, leur permettant d'être maîtres des contrôles notamment lors d'une sélection multi-utilisateurs.

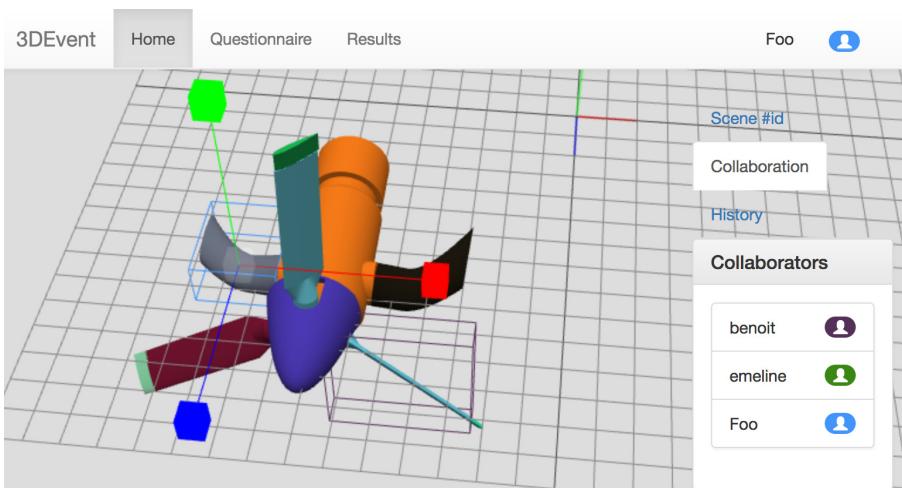
5.4. Comparaison entre l'expérimentation 1 et l'expérimentation 2



(a) Translation (environnement 3D) et visualisation de l'historique (panneau latéral)



(b) Rotation (environnement 3D) et outils pour la manipulation d'objet 3D (panneau latéral)



(c) Mise à l'échelle (environnement 3D) et liste des collaborateurs (panneau latéral)

FIGURE 5.1 – Interface utilisateur pendant une session collaborative (trois personnes)

5.4. Comparaison entre l'expérimentation 1 et l'expérimentation 2

The screenshot shows the Event Store Stream Browser interface. At the top, there is a green header bar with the Event Store logo and navigation links: Dashboard, Stream Browser (which is highlighted in blue), Projections, Query, Competing Consumers, Admin, Users, and Log Out. Below the header, the title "Event Stream 'scene-turbine'" is displayed, along with two buttons: "Edit ACL" and "Back". Underneath the title, there are four small buttons: "self", "first", "previous", and "metadata". The main area is a table with the following data:

Event #	Name	Type	Created Date	
19	19@scene-turbine	MeshRotatedEvent	2017-03-12 18:18:52	JSON
18	18@scene-turbine	MeshTranslatedEvent	2017-03-12 18:18:52	JSON
17	17@scene-turbine	MeshAddedToSceneEvent	2017-03-12 18:18:52	JSON
16	16@scene-turbine	MeshScaledEvent	2017-03-12 18:18:52	JSON
15	15@scene-turbine	MeshRotatedEvent	2017-03-12 18:18:52	JSON
14	14@scene-turbine	MeshTranslatedEvent	2017-03-12 18:18:52	JSON
13	13@scene-turbine	MeshAddedToSceneEvent	2017-03-12 18:18:52	JSON
12	12@scene-turbine	MeshScaledEvent	2017-03-12 18:18:52	JSON
11	11@scene-turbine	MeshRotatedEvent	2017-03-12 18:18:52	JSON
10	10@scene-turbine	MeshTranslatedEvent	2017-03-12 18:18:52	JSON

FIGURE 5.2 – Persistance long-terme (Event Store[®]), base de données/outil de monitoring

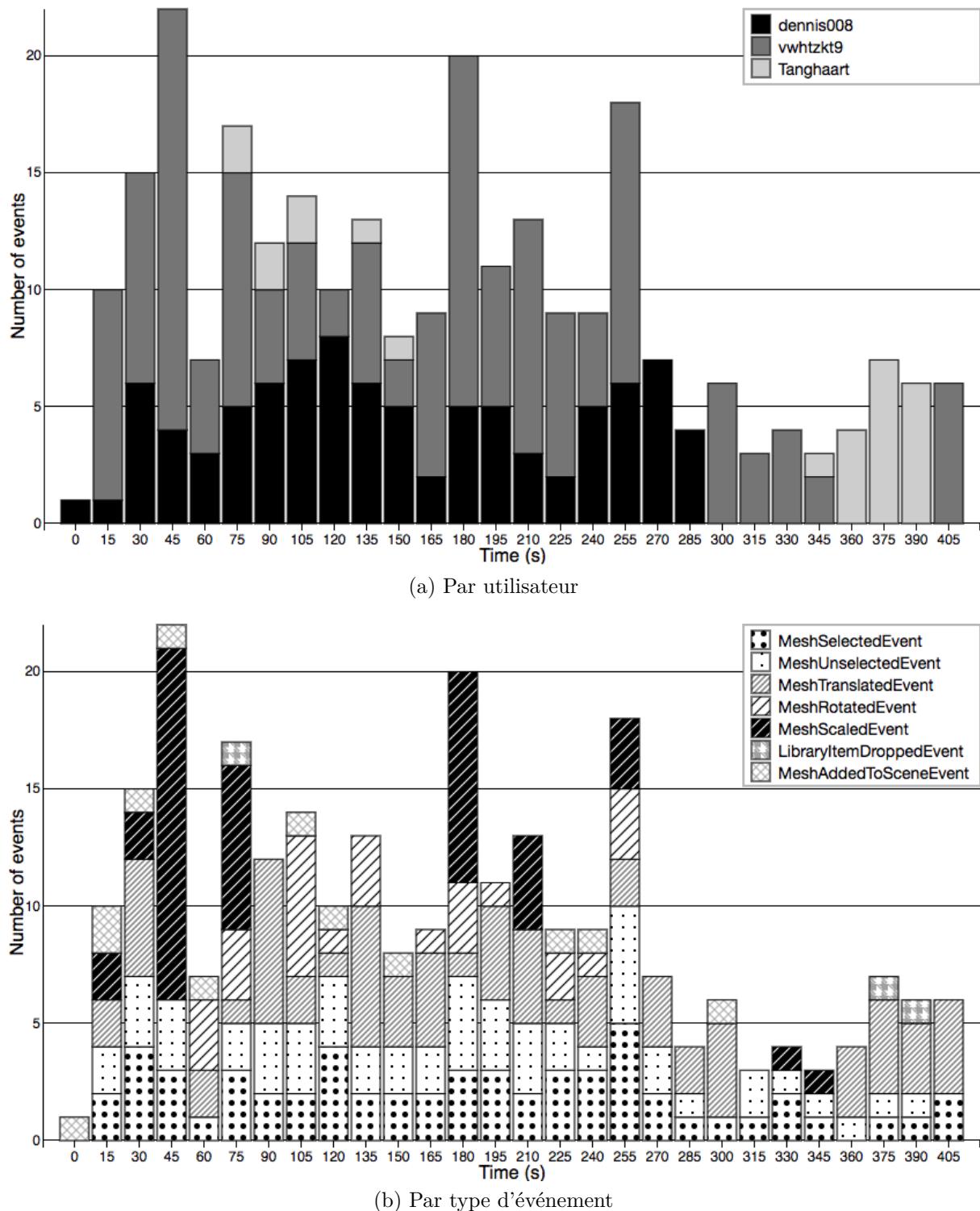


FIGURE 5.3 – Résumé d'une session collaborative au cours du temps

5.4. Comparaison entre l'expérimentation 1 et l'expérimentation 2

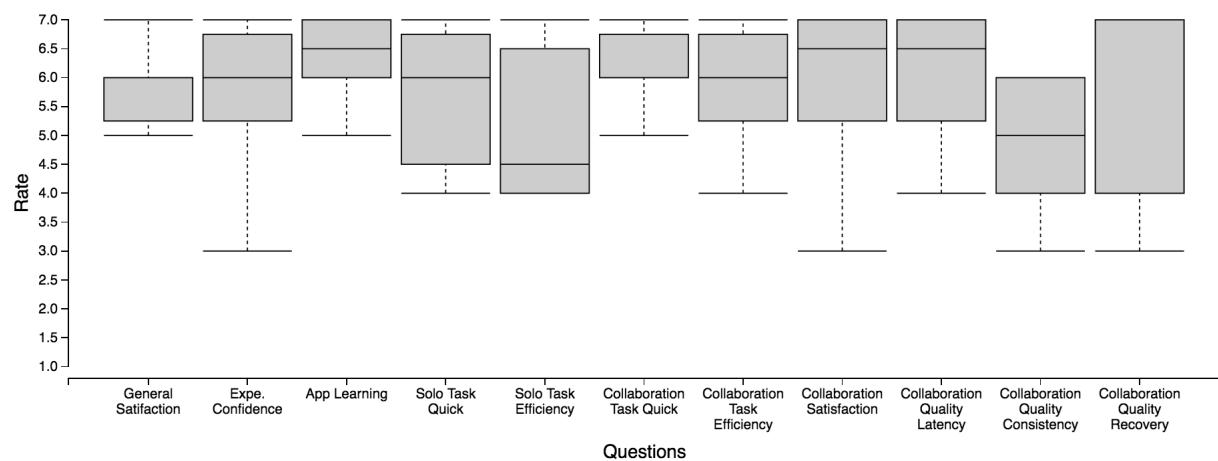


FIGURE 5.4 – Résultats des questionnaires collectés

Chapitre 6

Conclusion

Cette thèse s'inscrit dans le contexte la visualisation et de la manipulation d'objets 3D dans un **EVC** sur le web. Ce sujet intègre de nombreux verrous concernant différents domaines que la collaboration a tendance à réunir : gestion de la cohérence dans un environnement collaboratif massif en **P2P**, architecture de communication efficace pour le passage à l'échelle et la résilience, modèle de données dédiée au métier de la modélisation 3D collaborative.

Pour répondre aux différentes questions de recherche posées en introduction de ce manuscrit, plusieurs contributions ont été proposées, implantées et soumises à des expérimentations.

La **QR 1** concerne l'architecture réseau qu'un **EVC 3D** doit proposer pour pouvoir s'adapter à une gestion efficace, robuste et temps réel des données dans un environnement web. En effet, pour pouvoir collaborer les utilisateurs ont besoin d'une architecture de communication qui les relie de manière transparente intégrant un modèle de cohérence des données.

L'architecture de communication au sein de l'**EVC 3D** a été l'objet de deux contributions successives, la seconde s'inspirant des verrous issus de la première. L'architecture hybride à maillage complet se compose d'éléments simples. Seuls les navigateurs des utilisateurs participent au réseau **P2P**. Les données correspondent à l'état des objets manipulés. Les modifications sont transférées sous la forme d'un différentiel d'état. Bien que la politique de connectivité ne permettent pas un passage à l'échelle efficace, la cohérence des données sur le réseau est garantie de manière forte et pessimiste respectivement par le protocole de transport est configuré comme fiable et ordonnés, et par le mécanisme de verrouillage afin d'éviter l'édition concurrente d'objets 3D.

Partant de ces observations, la seconde contribution liées à l'architecture de communication repose sur le paradigme événementiel. Une politique de connectivité plus lâche permet aux pairs de devenir relais des informations. La partie serveur est « descendue » au niveau de la couche **P2P** pour permettre un accès réparti des pairs producteurs de données (navigateurs) à la persistance des données via des pairs relais. L'ensemble des pairs partage un journal d'événements qui est maintenu à jour et cohérent par à deux niveaux de vérification : au niveau du pair, avec un système de version lié à l'ordre causal puis au niveau de la grappe, avec un système de vote lors de la détection de conflit pour accepter ou rejeter collectivement un événement. La cohérence repose sur une stratégie optimiste avec des contraintes sur les règles métiers dont la permissivité peut être configurée. L'expérimentation 2, reposant sur cette seconde architecture, a montré que ce modèle optimiste est moins frustrant pour les utilisateurs. La couche **P2P** est enrichie de nouveaux pairs dédiés à la transmission des données qui permettent une dissémination plus

stable et rapide. Selon le nombre d'utilisateur, la montée en charge peut plus facilement être absorbée.

Dans la **QR 2** sont évoqués les verrous souvent rencontrés dans la modélisation 3D et particulièrement lorsque le volet collaboratif s'invite. Les fonctionnalités d'historique et de défaire / refaire une action sont fondamentales en modélisation 3D. Pour assurer un suivi de la création sans perdre l'intention de l'utilisateur de son action à sa visualisation, la modélisation sur le paradigme événementiel est adapté par ses capacités à permettre une forte traçabilité des changements qui se produisent au sein d'un système. La modélisation passe, dans un premier temps, par le suivi des bonnes pratiques énoncées par le **DDD** pour cerner les différents agrégats qui composent le domaine de la modélisation 3D collaborative. Puis, dans un second temps, deux principes sont combinés dans l'application cliente : (i) la séparation de la partie écriture de la partie lecture (**CQRS**) ; (ii) la création d'événements auto-descriptifs du changement qui vient de se produire (**ES**). Grâce à (i) impose plusieurs niveaux de vérification des données : de la saisie utilisateur à la validation les règles métier. Ainsi, le système, qui ne manipule que des événements immuables est garanti d'avoir des données valides. Côté lecture, les vues matérialisées de l'interface sont issues de projections faites à partir du flux d'événements traités par le système. Les vues sont spécifiques à l'activité, ce qui permet de présenter une interface orientée tâche plus proche des attentes métiers de l'utilisateur. La projection découpe les données de la base de données des données servant une ou plusieurs vues. En effet, son rôle d'intermédiaire est fournir les événements d'intérêts sous une forme dénormalisée pour les vues (mise en cache). Le point (ii) reprend principalement les avantages de l'**ES**, i.e. enregistrer la séquence d'événements qui a amenée à l'état courant de l'application. La fonctionnalité d'historique est implicitement présente dans ce patron, même ce qui pourrait être considéré comme une « annulation » par l'utilisateur est conservé sous la forme d'un événement de compensation. L'immuabilité des événements capte toutes les manipulations métier de l'utilisateur. Par conséquent, la reconstruction des agrégats du domaine permet d'avoir un état cohérent garanti par l'aspect fonctionnel de ce patron. C'est particulièrement important pour (i) améliorer la sensibilisation des collaborateurs à l'historique des objets manipulés ; (ii) faciliter le développement d'outils de surveillance pour observer, analyser et améliorer l'application ; (iii) et organiser des audits (exemple : respect d'une procédure).

Une partie de la contribution décrivant le modèle événementiel révèle une partie de la réponse à la **QR 3**. Le fait de déporter le patron **CQRS** en totalité sur le client évite que le client soit dépendant du serveur pour valider le contenu créé. L'utilisateur a accès localement aux événements qui le concernent

Annexe A

Ressources pour l'implantation et les expérimentations

A.1 Description des événements par agrégat dans 3DEvent

Tableau A.1 – 3DEvent : résumé des événements par agrégats

événement	Nommage	Description
Agrégat Scène		
Scène créée	sceneCreated	Une scène a été créée
Scène supprimée	sceneRemoved	Une scène a été supprimée
Scène renommée	sceneRenamed	Une scène a été renommée
Agrégat Maillage		
Maillage ajouté	meshAdded	Un maillage a été ajouté dans la Scène à partir d'une géométrie de la bibliothèque
Maillage déposé	meshDropped	Un maillage a été déposé dans l'environnement 3D de la Scène à partir d'une géométrie de la bibliothèque
Maillage supprimé	meshRemoved	Un maillage a été supprimé de la Scène
Maillage translaté	meshTranslated	Un maillage a subit une translation dans la Scène
Maillage pivoté	meshRotated	Un maillage a subit une rotation dans la Scène
Maillage mis à l'échelle	meshScaled	Un maillage a subit une homothétie dans la Scène
Agrégat Géométrie		
Géométrie importée dans la bibliothèque	geometryImported	Une géométrie est créée à partir d'un fichier importé par un utilisateur et ajoutée à la bibliothèque de la Scène
Agrégat Utilisateur		
Utilisateur créé	userCreated	Un utilisateur a été créé dans l'application
Scène rejointe par utilisateur	userJoinedScene	Un utilisateur a rejoint une scène
Scène quittée par utilisateur	userLeftScene	Un utilisateur a quitté une scène
Nom modifié	usernameChanged	Un utilisateur a modifié son nom
Couleur modifiée	colorChanged	Un utilisateur a modifié son code couleur

A.2 Messages réseaux pour la synchronisation des Event Stores

Tableau A.2 – Type de messages lors de la synchronisation

Message	Description
STREAM_SYNC_ASK	Demande de synchronisation d'un <i>stream</i>
CHUNK	Réception d'une donnée <i>chunk</i>)
READY_ASK	Prêt pour la démarrer la demande de données de sync.
READY	Prêt pour démarrer la réception de données de sync.
ALL_EVENTS_SYNC_ASK	Demande de toutes les données typées événement
EVENTS_SYNC	Réception de données (en cours de synchronisation)
META_DATA_ASK	Demande de métadonnées
META_DATA	Réception de métadonnées
SYNC	Réception de données (en cours de synchronisation)
EVENT	Réception d'une donnée typée événement
END_SYNC	Fin de la synchronisation

parler des tableaux

Tableau A.3 – Statut du noeud

Message	Description
ERROR	En erreur (désynchronisation)
READY	Prêt à recevoir des messages
META_DATA_ASK	En demande de métadonnées
META_DATA_RECEIVE	En réception de métadonnées
CLOSE	Déconnecté (connexion fermée)
RECEIVE_SYNC	En réception de données à synchroniser
CONNECTED	Connecté (connexion ouverte)
INIT	Initialisation
OK	Connecté et synchronisé
SEND_SYNC	En demande de synchronisation
END_SYNC	Synchronisation terminée

A.3 Experimentation 1 materials

A.3.1 Description des modèles utilisés

Tableau A.4 – Model descriptions for the experiments

Experiment	objects	size	users
Wind turbine	6	1.0 MB	2
Pick up	8	1.3 MB	4
Castle from <i>server</i>	35	1.3 MB	4
Castle from <i>peer</i>	35	1.3 MB	4

A.3.2 Questionnaire pour l'enquête utilisateur

Did you reach the goal or are you satisfied with the result ?

- Type of external communication channel used ?
- Collaboration satisfaction ?
- User interface quality : tools ? Object manipulation ? Global ?
 - Practice : difficulty or frustration ?
 - Collaborative rendering (latency, consistency, recovery) ?

A.4 Expérimentation 2

A.4.1 Description des modèles utilisés

Tableau A.5 – Modèles utilisés durant l'expérimentation

Modèle	Nombre de parties	Nombre de triangles	Taille totale
Rotor	10	62k	4Mo
Camera box	12	67k	5Mo
Car	16	170k	8Mo
Living room	16	200k	9Mo

A.4.2 Questionnaire pour l'enquête utilisateur

Seven-points scale questions from 1 (don't agree) to 7 (agree) :

- Did you enjoy this ?

- After trial, I was confident to do object manipulation in 3D virtual environment ?
- App learning : I am satisfied with the ease of use of the application
- App learning : It was easy to learn the tool
- Solo : I completed the task quickly
- Solo : I completed the task efficiently
- Collaboration : I contributed to complete the task quickly
- Collaboration : I contributed to complete the task efficiently
- In general, I am satisfied with the collaboration experience
- In general, The collaboration quality was pleasant in terms of LATENCY
- In general, The collaboration quality was pleasant in terms of CONSISTENCY
- In general, The collaboration quality was pleasant in terms of RECOVERY

Other questions :

- General efficiency is improved with the number of users ?
- General speed is improved with the number of users ?
- I would qualify this application : Non interactive/Interactive/Near real-time/Real-time
- Negative/positive/comments feedbacks

