Architecture événementielle pour les environnements virtuels collaboratifs sur le web : application à la manipulation et à la visualisation d'objets en 3D

Caroline Desprat

▶ **To cite this version:**

Caroline Desprat. Architecture événementielle pour les environnements virtuels collaboratifs sur le web : application à la manipulation et à la visualisation d'objets en 3D. Modélisation et simulation. Université Toulouse le Mirail - Toulouse II, 2017. Français. NNT : 2017TOU20103 . tel-02137485

HAL Id: tel-02137485

https://theses.hal.science/tel-02137485

Submitted on 23 May 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 2 Jean Jaurès*

Présentée et soutenue le *01/12/2017* **par :**

CAROLINE DESPRAT

Architecture événementielle pour les environnements virtuels collaboratifs sur le web : Application à la manipulation et à la visualisation d'objets 3D

JURY

THIERRY DUVAL	Professeur IMT, IMT-Atlantique	Rapporteur
JEAN-PIERRE JESSEL	Professeur des Universités, Université de Toulouse	Co-directeur
GUILLAUME LAVOUÉ	Maître de Conférences - HDR, INSA-Lyon	Rapporteur
HERVÉ LUGA	Professeur des Universités, Université de Toulouse	Directeur
GÉRALDINE MORIN	Maître de Conférences - HDR, ENSEEIHT	Examinatrice
NANCY RODRIGUEZ	Maître de Conférences, Université Montpellier	Examinatrice

---

École doctorale et spécialité :
MITT : Image, Information, Hypermédia
Unité de Recherche :
Institut de Recherche en Informatique de Toulouse (UMR 5505)
Directeur(s) de Thèse :
Hervé LUGA et *Jean-Pierre JESSEL*
Rapporteurs :
Thierry DUVAL et *Guillaume LAVOUÉ*

Résumé

L'évolution technologique du web durant ces dernières années a favorisé l'arrivée d'environnements virtuels collaboratifs pour la modélisation 3D à grande échelle. Alors que la collaboration réunit dans un même espace partagé des utilisateurs distants géographiquement pour un objectif de collaboration commun, les ressources matérielles qu'ils apportent (calcul, stockage, 3D ...) avec leurs connaissances sont encore trop rarement utilisées et cela constitue un défi. Il s'agit en effet de proposer un système simple, performant et transparent pour les utilisateurs afin de permettre une collaboration efficace à la fois sur le volet computationnel mais aussi, bien entendu, sur l'aspect métier lié à la modélisation 3D sur le web. Pour rendre efficace le passage à l'échelle, de nombreux systèmes utilisent une architecture réseau dite "hybride", combinant client serveur et pair-à-pair. La réplication optimiste s'adapte bien aux propriétés des ces environnements répartis : la dynamicité des utilisateurs et leur nombre, le type de donnée traitées (3D) et leur taille.

Cette thèse présente un modèle pour les systèmes d'édition collaborative en 3D sur le web. L'architecture cliente (3DEvent) permet de déporter les aspects métiers de la 3D au plus près de l'utilisateur sous la forme d'évènements. Cette architecture orientée événements repose sur le constat d'un fort besoin de traçabilité et d'historique sur les données 3D lors de l'assemblage d'un modèle. Cet aspect est porté intrinsèquement par le patron de conception event-sourcing. Ce modèle est complété par la définition d'un intergiciel en pair-à-pair. Sur ce dernier point, nous proposons d'utiliser la technologie WebRTC qui présente une API familière aux développeurs de services en infonuagique. Une évaluation portant sur deux études utilisateur concernant l'acceptance du modèle proposé a été menée dans le cadre de tâches d'assemblage de modèles 3D sur plusieurs groupes d'utilisateurs.

Mot clés : environnement virtuel collaboratif, réseau pair-à-pair, WebRTC, web 3D, conception 3D, traitement réparti évènementiel, architecture hybride, event-sourcing.

Abstract

Web technologies evolutions during last decades fostered the development of collaborative virtual environments for 3D design at large scale. Despite the fact that collaborative environments gather in a same shared space geographically distant users in a common objective, the hardware ressources of their clients (calcul, storage, graphics ...) are often underused because of the challenge it represents. It is indeed a matter of offering an easy-to-use, efficient and transparent collaborative system to the user supporting both computationnal and 3D design visualisation and business logic needs in heterogeneous web environments. To scale well, numerous systems use a network architecture called "hybrid", combining both client-server and peer-to-peer. Optimistic replication is well adapted to distributed application such as 3D collaborative envionments : the dynamicity of users and their numbers, the 3D data type used and the large amount and size of it.

This document presents a model for 3D web-based collaborative editing systems. This model integrates 3DEvent, an client-based architecture allowing us to bring 3D business logic closer to the user using events. Indeed, the need of traceability and history awareness is required during 3D design especially when several experts are involved during the process. This aspect is intrinsec to event-sourcing design pattern. This architecture is completed by a peer-to-peer middleware responsible for the synchronisation and the consistency of the system. To implement it, we propose to use the recent web standard API called WebRTC, close to cloud development services know by developers. To evaluate the model, two user studies were conducted on several group of users concerning its responsiveness and the acceptance by users in the frame of cooperative assembly tasks of 3D models.

Keywords : collaborative virtual environment, peer-to-peer network, WebRTC, Web 3D, 3D design, distributed event-based system, hybrid architecture, event-sourcing.

Remerciements

Je tiens tout d'abord à remercier les membres du jury pour avoir accepté de participer à ce comité, en particulier, les rapporteurs pour leurs commentaires pertinents et constructifs lors des pré-rapports.

Ensuite, je tiens à remercier Hervé Luga, mon directeur de thèse, et Jean-Pierre Jessel, mon co-encadrant de thèse, pour l'encadrement qu'ils ont exercé durant ces trois ans (et onze mois), en m'accordant une grande confiance dans le choix et la réalisation du sujet de cette thèse. Hervé, qui depuis le stage de Master m'a suivie et donné la possibilité d'ouvrir ma recherche à d'autres champs disciplinaires. Jean-Pierre, qui a su distiller les opportunités et les ressources au cours de ma thèse pour voir au delà de la vie de laboratoire.

Je aussi le sentiment d'avoir été un objet test de l'université fédérale de Toulouse entre mon université d'inscription et d'enseignement (UT2J), celle où se situait mon bureau (UT1-C) et celle des projets et des séminaires (UPS). Cela m'a permis de rencontrer de nombreuses personnes, collègues et amis formidables que je tiens également à remercier pour leur patience ainsi que pour les longs et passionnés échanges que nous avons eus. Je pense en particulier à la ME310, où l'ambiance chaleureuse (surtout l'été), fut le berceau de belles amitiés.

Les ami · e · s. Toi qui a relu un papier la veille pour le lendemain ou cette thèse sans n'y rien connaître, ou encore toi qui ne m'a pas posé 100 fois la question « alors c'est prévu pour quand la soutenance ? », merci.

Papounet, toujours présent même à distance. Les bons petits plats, les petits sms, les voyages... Toujours sensibles, parfois silencieux, parfois musicaux, ces encouragements m'ont toujours donné des appuis sur lesquels me reposer. Merci.

Julia, sista d'amour, la motivation derrière cette thèse te doit beaucoup et moi aussi. En partageant les expériences de thèse de l'une et l'autre nous avons appris beaucoup sur chacune. J'ai trouvé dans ma sœur chérie et adorée une femme brillante, disponible et spontanée. Merci.

Benoît, je pense que tu n'envisageais pas cette aventure de cette manière. Moi non plus. Toujours là, patient, à m'écouter, me proposer des idées et à me soutenir. Ces moments de thèses partagés ensemble sont des morceaux de vie uniques que je suis heureuse d'avoir partagés avec toi. Merci.

À maman,

Table des matières

Résumé i

Remerciements ii

Table des matières vii

Table des figures ix

Liste des tableaux xi

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 . . . 6
  - 1.1.4 Les architectures orientées événements pour la collaboration . . . 8
- 1.2 Problématique . . . 12
- 1.3 Contributions . . . 14
  - 1.3.1 Contributions théoriques . . . 15
  - 1.3.2 Contributions pratiques . . . 15
- 1.4 Organisation du manuscrit . . . 16

2 État de l'art 17
- 2.1 Modélisation 3D collaborative sur le web . . . 18
  - 2.1.1 Collaboration et concurrence en 3D . . . 18
  - 2.1.2 ... sur le web : HTML5, web 3D et au-delà . . . 20
- 2.2 Communication en temps-réel . . . 25
  - 2.2.1 Les principaux protocoles de transport . . . 29
  - 2.2.2 WebRTC : le web hypermedia en P2P . . . 31
  - 2.2.3 Quel protocole dans quel EVC3D ? . . . 34
- 2.3 Les systèmes distribués orientés événements pour la collaboration . . . 36
  - 2.3.1 Introduction . . . 36
  - 2.3.2 Systèmes Publish-Subscribe . . . 36
  - 2.3.3 Domain Driven Design . . . 40
  - 2.3.4 Command Query Responsability Segregation . . . 43
  - 2.3.5 Event Sourcing . . . 44
- 2.4 Conclusion . . . 48

3 Contributions **51**
3.1 Introduction . . . 52
3.2 Modèle événementiel pour l'intégration du domaine 3D dans les EVC . . . 53
3.2.1 Modèle général . . . 55
3.2.2 Adaptation des patrons Event Sourcing (ES) et Command Query Responsability Segregation (CQRS) . . . 58
3.2.3 Journal des événements . . . 60
3.2.4 Flexibilité de la visualisation . . . 63
3.2.5 Cohérence éventuelle en CQRS . . . 63
3.2.6 Bilan . . . 65
3.3 Architecture de communication hybride . . . 66
3.3.1 Éléments constitutifs d'un pair . . . 69
3.3.2 Architecture hybride « orientée états » . . . 72
3.3.3 De l'état à l'événement . . . 76
3.3.4 Architecture de communication hybride « orientée événements » . . . 77
3.3.5 Bilan comparatif . . . 85
3.4 Conclusion du chapitre . . . 85

4 Prototypes d'application web pour la conception 3D collaborative **87**
4.1 Introduction . . . 88
4.2 3DState : une preuve de concept orientée états . . . 90
4.2.1 Implantation des composants pour la communication . . . 90
4.2.2 Interface de 3DState . . . 91
4.2.3 Gestion de la session . . . 91
4.2.4 Bilan critique . . . 92
4.3 3DEvent : une approche découplée orientée événements . . . 94
4.3.1 Intergiciel pour l'échange de 3D par événements . . . 94
4.3.2 Interface orientée tâches . . . 99
4.3.3 Sélection fantôme . . . 101
4.3.4 Bilan . . . 103
4.4 Conclusion du chapitre . . . 105

5 Études utilisateurs sur l'assemblage collaboratif d'objets 3D **107**
5.1 Cas d'étude : Assemblage collaboratif d'objets 3D dans un environnement web 108
5.2 Expérimentation 1 : preuve de faisabilité . . . 109
5.2.1 Présentation de l'expérimentation 1 . . . 109
5.2.2 Résultats et discussion . . . 111
5.2.3 Conclusion de l'expérimentation 1 . . . 112
5.3 Expérimentation 2 : intégration du framework événementiel . . . 113
5.3.1 Présentation de l'expérimentation 2 . . . 113
5.3.2 Résultats et discussion . . . 115
5.3.3 Conclusion de l'expérimentation 2 . . . 120
5.4 Comparaison entre l'expérimentation 1 et l'expérimentation 2 . . . 121

6 Conclusion **123**
6.1 Discussion 124
6.2 Potentielles applications et autres utilisations 128
6.3 Perspectives 129

Bibliographie **131**

A Ressources pour l'implantation et les expérimentations **141**
A.1 Description des événements par agrégat dans 3DEvent 141
A.2 Messages réseaux pour la synchronisation des Event Stores 143
A.3 Experimentation 1 materials 144
A.3.1 Description des modèles utilisés 144
A.3.2 Questionnaire pour l'enquête utilisateur 144
A.4 Expérimentation 2 144
A.4.1 Description des modèles utilisés 144
A.4.2 Questionnaire pour l'enquête utilisateur 144

Glossaire **147**

Acronymes **149**

Table des figures

1.1 Place de la contribution dans les différents champs de recherche 13

2.1 Déclaratif vs Impératif en 2D et en trois dimensions (3D) sur le web 23
2.2 Support de WebGL 1 (2014-2017) et WebGL 2 (2016-2017) 25
2.3 Pile des protocoles IP: TCP vs UDP 32
2.4 STUN, TURN et signalisation 34
2.5 Architecture Publish-Subscribe 37
2.6 Illustration du patron Domain Driven Design (DDD) et de ses artefacts . . . 41
2.7 Architecture en 4 couches du DDD (gauche) en miroir avec l'architecture CQRS (droite) 43
2.8 Transaction en ES 45
2.9 Snapshot en ES 46

3.1 Illustration des notions d'événements du domaine, des types d'événements, et des instances d'événements. 56
3.2 Modèle de l'architecture client dans 3DEvent 59
3.3 Représentation d'un journal d'événement 61
3.4 Théorème CAP et les algorithmes de compromis 64
3.5 Architecture hybride tripartite : serveur, persistance et les pairs. 68
3.6 Composants de chaque pair dans 3DEvent (point de vue réseau) 70
3.7 Architecture de communication « orientée événements » 78
3.8 Illustration de l'architecture de communication « orientée événements » avec 4 Instances Web et 2 Instances Serveur 79
3.9 Protocole de connexion au réseau d'instance 3DEvent 81

4.1 Interface de 3DState 92
4.2 Diagramme de séquence de la gestion de la session dans une architecture « orientée état » 93
4.3 Exemple d'agrégats 98
4.4 Onglets du panneau latéral de l'interface 100
4.5 Flux de la collaboration dans le framework 3DEvent entre 3 utilisateurs . . . 102
4.6 Illustration de la sélection fantôme dans l'environnement 3D 104

5.1 Interface utilisateur pendant une session collaborative (trois personnes) . . . 116
5.2 Persistance long-terme (Event Store®), base de données/outil de monitoring . 117
5.3 Résumé d'une session collaborative au cours du temps 118
5.4 Résultats des questionnaires collectés 119

Liste des tableaux

2.1 Solutions conventionnelles pour les problèmes collaboratifs [Yu *et al.*, 2016]. . 19
2.2 Encodage des données transmises 23
2.3 Aperçu des procotoles de transport 31

3.1 Événements de l'agrégat Maillage (extrait de Tab. A.1) 58
3.2 Récapitulatif des deux approches d'architecture de communication 86

4.1 Résumé des choix techniques pour 3DState 94
4.2 Récapitulatif des choix techniques de 3DEvent 104

A.1 3DEvent : résumé des événements par agrégats 142
A.2 Type de messages lors de la synchronisation 143
A.3 Statut du nœud 143
A.4 Modèles utilisés pendant l'expérimentation 1 144
A.5 Modèles utilisés durant l'expérimentation 2 144

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 6
1.1.4 Les architectures orientées événements pour la collaboration 8
1.2 Problématique 12
1.3 Contributions 14
1.3.1 Contributions théoriques 15
1.3.2 Contributions pratiques 15
1.4 Organisation du manuscrit 16

1.1 Contexte

Le besoin de collaboration et de manipulation en temps réel d'objets en 3D ainsi que leur contrôle de version a été très tôt une raison de la dissémination des outils de 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'héritage culturel ou la conception d'objets manufacturés, 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 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 comprend également 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 récursif[1] 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 dans des environnements et des systèmes complexes. Les imbrications de chaque domaine et la trans-

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].

diciplinarité 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.

Vers la fin des années 2000, deux classifications ont été retenues concernant la collaboration [Bauwens, 2011]. 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 libre (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 contexte de 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 au code source ouvert)

— *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 supplémenter 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'intérêt 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 encouragée 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 distants 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 (multi-utilisateurs). Un EVC 3D permet à plusieurs utilisateurs d'interagir les uns avec les autres en (quasi) temps 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], un

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 développement 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 travaillent sur des ressources réparties sur le réseau. 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, objet 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 principale 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 pour plusieurs domaines, notamment l'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'inhabituel 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 Larousse[2]. 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 instantané, 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.

2. Larousse - événement : http://www.larousse.fr/dictionnaires/francais/événement/31839. Consulté le 16/10/2017

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*) [3], 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.

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*) [4] dans des activités. Ces systèmes sont présents dans la littérature depuis les prémices de la technologie web [Bentley et Wakefield, 1997, Steinfield *et al.*, 1999, You et Pekkola, 2001]. Ces propositions

3. 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.

4. 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

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 un système de communication restreint ; la présence 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 intégrant des mécanismes de sensibilisation [5] à plusieurs niveaux : communication, disponibilité, activité. Ces derniers sont directement implémentés dans l'intergiciel 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.* Dans un système décentralisé, l'information sur laquelle la sensibilisation est construite est répartie sur les sites des différents pairs. Dans les approches centralisées, le traitement des événements est effectué sur le serveur central qui, en lien avec la base de données, stocke les événements et l'historique de l'application puis les distribue aux clients via l'interface de requête permettant d'extraire les informations d'intérêt. Dans le collecticiel P2P, le stockage, le traitement et les requêtes d'événements doivent être faits de manière répartie pour tirer parti de la distribution des données.

— *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 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 gé-

5. Traduction du terme *awareness* dans cette thèse : indications ambiantes dans l'environnement, suite à des notifications ponctuelles.

nériques possibles, i.e. qu'ils puissent être traités de la même manière par tous les composants, 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 limités.

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 situations accidentelles (effets de bords, conflits). 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 simulations distribuées, 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 collaboration 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 des aspects du métier dans le processus de la manipulation et de la visualisation des objets 3D.

Les EDAs sont également un socle pour l'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. Elles s'appuient sur l'intégration de la sémantique métier aux données proposée par la modélisation des événements.

1.2 Problématique

En s'intéressant à ce que chaque acteur du système peut apporter à la collaboration, la problématique de cette thèse s'organise autour des différentes manières d'engager les ressources à disposition lors de la collaboration sur le web. Cette thèse se situe à l'interface de trois champs de recherche (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 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.

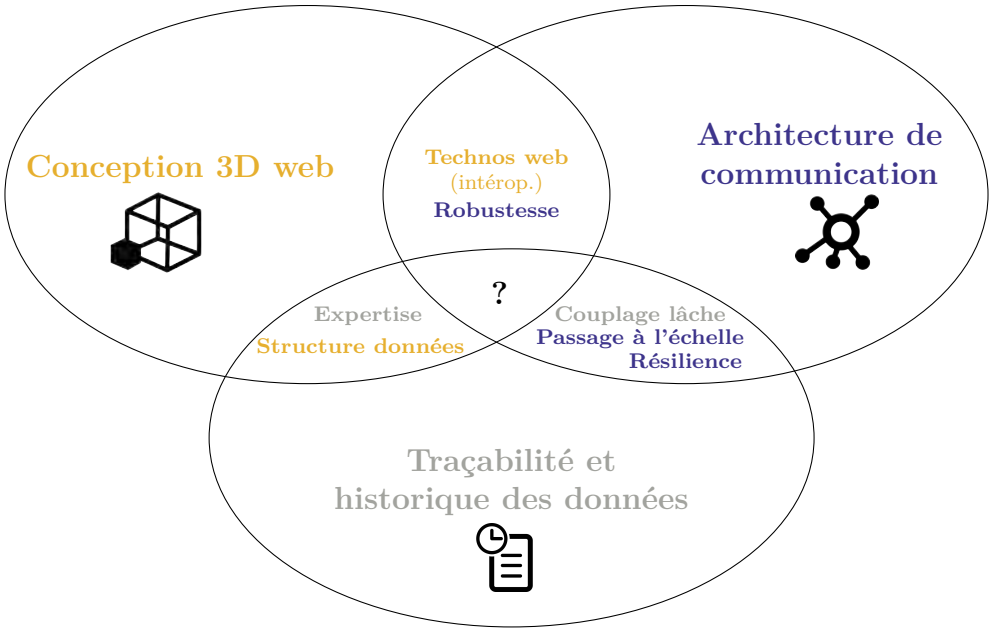


FIGURE 1.1 – Place de la contribution dans les différents champs de recherche

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 disponibles, telles que les clients (navigateurs web) sur lesquels les utilisateurs manipulent les modèles 3D, la gestion de la charge est plus flexible car les ressources augmentent avec le nombre de clients. Cela fournit également plus d'autonomie aux utilisateurs car le système leur permet d'utiliser leurs ressources propres, ce qui n'est pas une pratique développée par le tout-infonuagique.

Le second champ de recherche s'intéresse à l'architecture de communication nécessaire dans un EVC 3D sur le web prenant en compte les contraintes d'un tel environnement distribué. Les aspects liés à la décentralisation du stockage et de la distribution sont également étudiés afin de favoriser le passage à l'échelle du système et l'utilisation optimale de toutes les ressources mises à disposition par les différents utilisateurs lors de la collaboration.

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 concernent principalement l'historique, la traçabilité de l'information et l'expertise embarquée dans le système.

L'objectif de cette thèse est triple : (i) s'appuyer sur 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 Quelles sont architecture réseaux permettant 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 métiers liées à la manipulation d'objets 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 métiers liées à la manipulation d'objets 3D ?

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

1.3 Contributions

Cette thèse expose plusieurs contributions conceptuelles et pratiques. Une des contributions consiste en la proposition d'un 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 des patrons de conception DDD, CQRS et ES qui sont très peu confrontés à ce domaine.

La seconde contribution de cette thèse concerne l'architecture de communication dans un environnement collaboratif web. Celle-ci doit permettre au système d'être résilient, léger et transparent pour l'utilisateur. L'architecture hybride proposée repose sur une combinaison de l'architecture client-serveur et l'architecture P2P. Cela tient à la nécessité de centralisation de l'information dans un contexte industriel et à la haute disponibilité des ressources que permettent les propriétés du P2P. Cette contribution est elle-même découpée en deux parties. La première expose une preuve de concept sur la base du modèle présenté dans [Desprat *et al.*, 2015b, Desprat *et al.*, 2015a] 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. La gestion des données dans un environnement de modélisation 3D est importante, notamment concernant la conservation de l'historique des scènes et

des objets 3D. Le paradigme événementiel, qui offre au système l'avantage d'être peu couplé, intègre ces informations naturellement en plus des informations métiers qui sont générées par les collaborateurs dans l'application. En séparant le traitement de vérification des données de celles à visualiser, l'implantation d'un intergiciel P2P est d'autant facilitée.

Les processus et les outils actuels sont insuffisants pour réellement supporter la modélisation collaborative 3D sur le web actuellement. Cela concerne notamment l'identification de problèmes spécifiques à la manipulation d'objets 3D et leur formulation du point de vue expert de l'utilisateur pour enfin être intégrer aux applications.

1.3.1 Contributions théoriques

- Introduction des concepts d'événements métiers liés à la 3D comme moyen d'interagir dans une scène 3D.
- Modèle événementiel pour la manipulation et la visualisation d'objets 3D de manière collaborative : proposition d'un modèle CQRS ES uniquement sur le client avec un historique fonctionnel et une gestion de la cohérence intégrée.
- Définition d'une architecture hybride (client serveur et P2P) adaptée à la distribution de contenus 3D dans le cadre d'une collaboration sur le web en temps réel :
  - proposition d'une architecture hybride « orientée états » : transmission par différentiels d'état avec une cohérence forte ;
  - proposition d'une architecture hybride « orientée événements » : transmission de notifications d'événements avec une cohérence éventuelle.

1.3.2 Contributions pratiques

- Définition d'un framework dédié à la modélisation 3D collaborative s'appuyant sur les principes et les technologies du web.
- Implantation du prototype 3DState comme concept applicatif pour la modélisation 3D collaborative.
- Implantation du prototype 3DEvent proposant :
  - une interface orientée tâches spécifique à l'assemblage 3D ;
  - un outil de monitoring de la collaboration ;
  - un intergiciel P2P web proposant, en plus du navigateur, l'intégration de nouveaux types de pairs pour améliorer la distribution lors de la collaboration.

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. De nouveaux modèles, outils et méthodes facilitant la mobilité et l'autonomie des utilisateurs sont nécessaires. 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 2 présente l'état de l'art dans la collaboration 3D en s'intéressant à la modélisation 3D collaborative sur le web, puis aux problématiques de temps réel dans ce contexte et enfin à l'apport des systèmes événementiels distribués pour la collaboration.

Ensuite, le chapitre 3 décrit les contributions conceptuelles concernant le modèle événementiel proposé dans cette thèse ainsi que les deux architectures de communications hybrides (par états et par événements) qui ont été modélisées pour créer et échanger en collaborant sur des modèles 3D de manière synchrone et répartie.

Dans le chapitre 4 l'implantation des contributions est décrite sous la forme de deux prototypes. Le premier (3DState) s'appuie sur l'architecture de communication « orientée états ». Le second (3DEvent) utilise, d'une part, le modèle événementiel pour traiter les événements directement sur le pair. L'interface orientée tâches que le modèle héberge permet aux utilisateurs de manipuler des objets 3D de manière autonome tout en respectant les règles métiers. D'autre part, l'architecture de communication « orientée événements » qui introduit l'intergiciel P2P, capable non seulement de disséminer les données mais également de gérer la cohérence de l'application de manière distribuée. L'intergiciel P2P s'adapte à différents types de clients et offre également une bonne traçabilité des données au sein du réseau.

Puis, le chapitre 5 présente les deux expérimentations qui ont été menées respectivement sur les deux prototypes développés. Chacune propose, sous la forme d'une étude utilisateur, d'évaluer le prototype sur la base du cas d'usage présenté dans la section 5.1 : les utilisateurs doivent effectuer des assemblages de manière collaborative à partir d'un modèle séparé en plusieurs pièces, réparties entre chacun des utilisateurs. Les résultats sont extraits principalement des questionnaires, mais également de l'outil de surveillance dans le cas de 3DEvent. Le détails des analyses est ensuite présenté.

Enfin, le chapitre 6 contient la conclusion de la thèse en répondant aux **QR** posées dans cette introduction et ouvre sur les potentielles applications des contributions portées par cette thèse ainsi que les perspectives de recherche qui en découlent.

Chapitre 2

État de l'art

Sommaire

2.1 Modélisation 3D collaborative sur le web 18
2.1.1 Collaboration et concurrence en 3D 18
2.1.2 ... sur le web : HTML5, web 3D et au-delà 20
2.2 Communication en temps-réel 25
2.2.1 Les principaux protocoles de transport 29
2.2.2 WebRTC : le web hypermedia en P2P 31
2.2.3 Quel protocole dans quel EVC3D ? 34
2.3 Les systèmes distribués orientés événements pour la collaboration 36
2.3.1 Introduction 36
2.3.2 Systèmes Publish-Subscribe 36
2.3.3 Domain Driven Design 40
2.3.4 Command Query Responsability Segregation 43
2.3.5 Event Sourcing 44
2.4 Conclusion 48

2.1 Modélisation 3D collaborative sur le web

2.1.1 Collaboration et concurrence en 3D

Réaliser une tâche ou une activité de manière collaborative peut permettre soit d'atteindre un objectif qui n'est pas réalisable par une seule personne (exemple : faire porter trente chopes de bières par une personne[1]), soit d'être plus efficace et plus rapide pour l'atteindre (exemple : faire porter trente chopes par trente personnes). 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) avec un support visuel manipulable, éditable et flexible (proposant différentes possibilités de visualisation). En se complétant, leurs activités peuvent 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 souhaités (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. Celles-ci 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 classés comme pessimistes ou optimistes (Tableau 2.1). Les solutions de type autoritaire (« *authoritative* ») sont comptées parmi les solutions pessimistes car elles empêchent 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 plus permissives, et ne s'empêchent pas la possibilité de manipulation concurrente menant potentiellement à l'occurrence 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 mecanism*) 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]

1. Guinness record : 29 chopes en une fois en 2017

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	Résolution de conflit temps réel pour les documents textuels
	Transformées opérationnelles 3D	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

est un éditeur collaboratif 3D sur le web dont la gestion de la concurrence repose sur la demande d'accès à un objet (ce qui verrouille l'objet pour les autres utilisateurs lorsqu'elle est accordée).

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 d'édition massive 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'intéressant à la cohérence dans un environnement distribué collaboratif concernent 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 agnostiques[2] 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 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

2. 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.

de valeurs de hachage (*hash*) signées. Elles sont mises à jour régulièrement de manière décentralisée (*decentralized mirroring*). Cela permet de 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, web 3D et au-delà

L'arrivée du web a bouleversé les usages liés à la collaboration sur des objets 3D. Les principes et les technologies 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), de BIM ou de CAO, 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, simplification, *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 du 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. Cette synchronisation n'a souvent qu'un objectif de sauvegarde dans l'infonuagique pour les modèles 3D créés qui peuvent éventuellement être visualiser à distance. Généralement, les logiciels spécialisés dans la modélisation 3D, contrairement aux jeux multijoueurs, sont traditionnellement dédiés à un utilisateur unique. Les utilisateurs ne sont donc pas habitués à une collaboration synchrone sur une même scène. Cela implique qu'ils ne sont pas forcément familiers des manipulations concurrentes ou encore d'une représentation visuelle des autres opérateurs dans le même espace 3D. La sensibilisation à la collaboration porte beaucoup de responsabilités dans ce contexte. 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 étaient conçus pour moins de cent utilisateurs simultanés et reposaient sur un ou plusieurs serveurs pour supporter ces utilisateurs. Ce schéma est expliqué par la volonté du fournisseur de service d'assurer une qualité de service et la sécurité du système. Contrairement aux jeux massivement multi joueurs, 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.

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, l'architecture client-serveur est très répandue et l'utilisation du protocole WebSocket est souvent favorisé 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 TinkerCAD. Ces

plateformes proposent en général un environnement 3D pour visualiser une scène composée d'objets 3D, une fonctionnalité (plus ou moins développée) défaire / refaire et une plateforme de diffusion des créations. Du plus riche au plus généraliste, voici une liste de services donnant un aperçu des plateformes de modélisation 3D sur le web associés à leur spécificité.

OnShape est un service de modélisation 3D très riche pour la CAO 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] qui, très similaire à Git, propose un arbre des modifications (dont le nœud racine correspond à la scène vierge) lié à la scène où chaque micro-version stocke uniquement les informations relatives à son parent : deux nœuds peuvent être issus du même parent menant à la création d'une nouvelle branche, i.e. nouvel espace de travail séparé. 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 que OnShape. Il s'oriente davantage vers des fonctionnalités avancées de rendu 3D avec lancer de rayons dans l'infonuagique. Les fonctionnalités d'historisation et de collaboration proposées dans Clara.io restent très basiques (seulement annuler / refaire) tout comme la plateforme de publication proposée.

Plateforme de publication d'objets 3D populaire, TinkerCAD se focalise sur l'accès à la conception d'objets 3D pour les néophytes. 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,Constructive Geometry Solid (CSG)) [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

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 illustre 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.

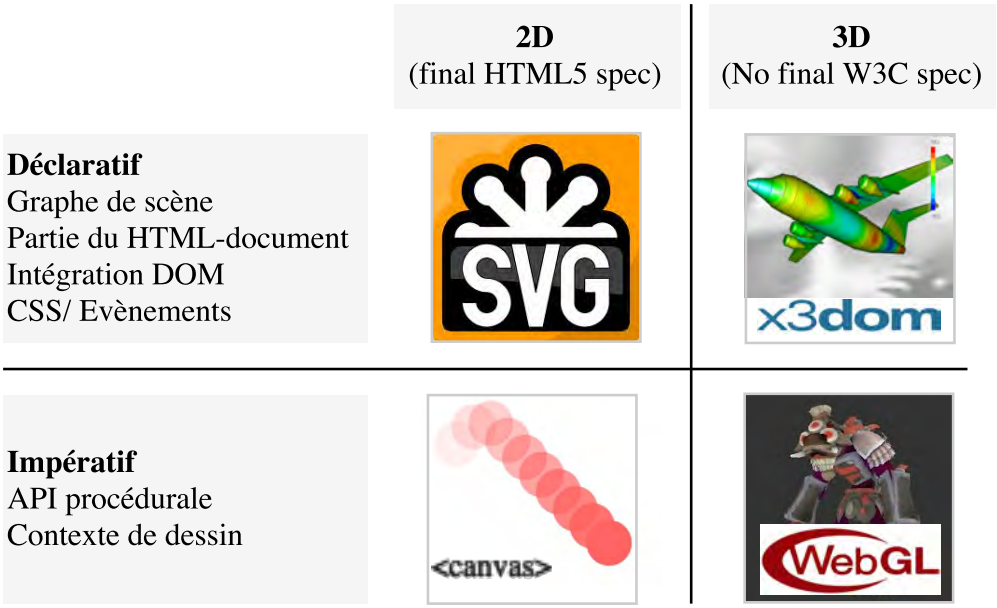


FIGURE 2.1 – Déclaratif vs Impératif en 2D et en 3D sur le web

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-compatible 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.

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 11 [3], Edge 14 [3] 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 Mini [4]. La spécification WebGL 2.0 [Khronos, 2016] basée sur OpenGL ES 3.0 [Khronos, 2008], déjà publiée en tant

3. Le contexte WebGL est accessible depuis « `experimental-webgl` » au lieu de « `webgl` ».
4. Données issues de `http://caniuse.com/#search=webgl`, consulté le 16/10/2017.

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

que brouillon, est en phase expérimentale [5]. La Figure 2.2 montre l'évolution du support de WebGL 1 et WebGL 2 durant ces dernières années sur les différentes plateformes possédant un navigateur web [6].

2.2 Communication en temps-réel

La communication dans un EVC est primordiale pour permettre aux collaborateurs d'échanger sur les données. 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'expression temps réel est largement utilisée pour décrire des applications requérant une forme de temps de réponse rapide et réactif pour que l'utilisateur ait une bonne expérience. L'exigence vis-à-vis des temps de réponse est importante, notamment dans le cadre des EVCs, 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 deux secondes ou plus à char-

5. http://caniuse.com/#feat=webgl2. Consulté le 16/10/2017
6. 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 16/10/2017.

ger [7]. 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. [8]

— *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.

— *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 la précision des calculs souhaitée et le temps de transmettre le résultat 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 ...), 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 par exemple dans un Environnement Virtuel (EV) ou un jeu sérieux multi-utilisateur (*serious game*), les participants interagissent en (quasi) temps réel, alors qu'ils sont situés à différents endroits géographiques. Le développement d'un EVC 3D partage les problématiques de conception et d'implantation réseau des systèmes distri-

7. www.akamai.com

8. 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.

bués, la gestion des données concurrentes, la visualisation 3D et l'ergonomie des interfaces 3D proposant des fonctionnalités collaboratives.

Dans le monde du réseau, le protocole de communication en temps-réel Real-Time Transport Protocol (RTP) 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 brutes (exemple : données 3D). Pour cette raison, leur utilisation est limitée dans le cadre de la modélisation 3D collaborative.

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.

Transport 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.

La fiabilité dans une communication est une propriété habituellement recherchée, cependant elle implique par ailleurs que le canal de communication connaisse 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 importantes (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. La connexion est rapide et stable dans ce contexte ; 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.

Transport Ordonné versus Désordonné Le routage des paquets de l'expéditeur vers le destinataire peut être différent selon l'état du réseau. La réaction des réseaux face à la congestion consiste à modifier les tables de routage afin que les paquets évitent le nœud 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 communica-

tions 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'Edition 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 de deux rotations consécutives se traduit par l'opération non-commutative de la multiplication de deux matrices 3D.

2.2.1 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 « somme de contrôle » (*checksum*) afin de vérifier les données dans les cas d'erreurs (mauvais signal, routeur défectueux). Si la somme est incorrecte, la donnée est écartée, 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 établi 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 somme de contrôle pour détecter les potentielles erreurs dans un paquet. Cependant, cette fonctionnalité est optionnelle dans IPv4 qui rapporte 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 gère 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 un accusé de réception sélectif. Cet accusé 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 donnée dans le nouveau paquet.

Le champ somme de contrôle dans l'entête 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 procotoles de transport

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

2.2.2 WebRTC : le web hypermedia en P2P

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 World Wide Web Consortium (W3C) et l'Internet Engineering Task Force (IETF)®. 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 WebRTC [9], plus d'un milliard d'utilisateurs ont utilisé la technologie *open source* WebRTC. Ericsonn 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) [10], des systèmes de partage de fichiers [11] et autres applications (capture d'écran, réalité augmentée,

9. https://webrtc.org
10. WebRTCWorld en liste un peu plus de 140 http://www.webrtcworld.com/webrtc-list.aspx. Consulté le 07/07/2017.
11. WebTorrent https://github.com/feross/webtorrent. Consulté le 04/09/2017.

jeux) [12]. 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'extrémité (*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).

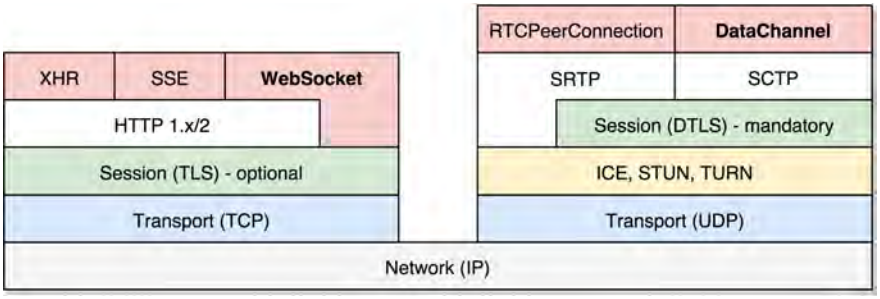


FIGURE 2.3 – Pile des protocoles IP : TCP vs UDP

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

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

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 mecanism* 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 (Interactive Connectivity Establishment (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 envoyés 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 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,

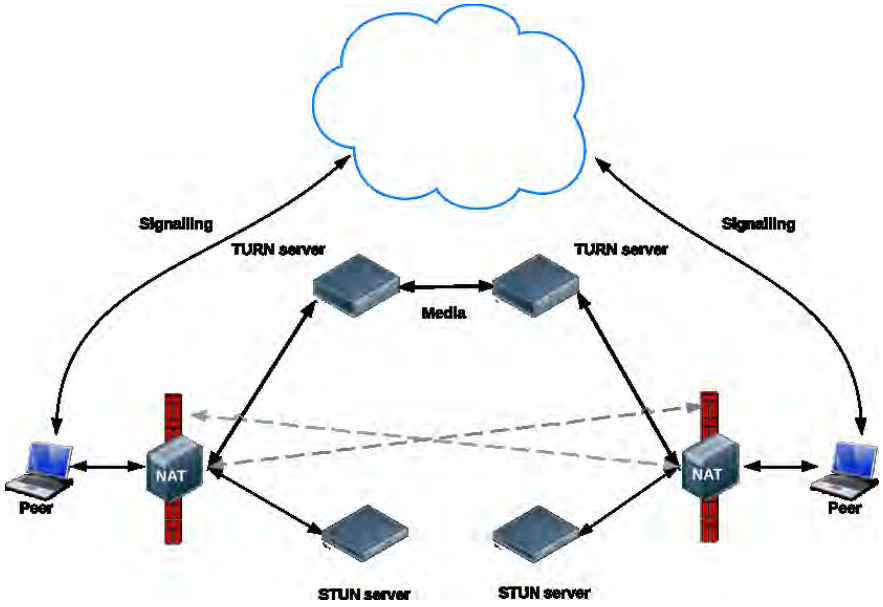


FIGURE 2.4 – STUN, TURN et signalisation

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'un NAT dynamique impose des restrictions sur l'adresse de destination. Dans ce cas, le STUN est remplacé par l'utilisation d'un serveur Traversal Using Relays around NAT (TURN). Le TURN est considéré comme une extension du STUN et fait office de proxy. Une fois que le client est parvenu à récupérer une adresse IP, il peut participer à l'initialisation d'une connexion WebRTC (signalisation). L'utilisation de protocoles tels que ICE avec son ICE Framework permet de faciliter la mise en relation des clients derrière un NAT. Le ICE Framework choisit le chemin le plus court entre les deux pairs et va utiliser le protocole adapté pour établir la connexion (STUN d'abord puis en cas d'échec, TURN).

2.2.3 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 dans un EV, un jeu sérieux multi-utilisateur, 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.

Comme l'indique [Roberto *et al.*, 2014], le protocole le plus souvent rencontré dans les EVC 3D est l'UDP. Cette prépondérance s'explique selon deux facteurs. Premièrement, dans

des réseaux où l'accès est fiable, 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 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 gérées 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 [Zhu *et al.*, 2011] 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. 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. 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. 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

Les systèmes orientés événements (EDA) sont très populaires lorsqu'il s'agit de systèmes collaboratifs [Helmer *et al.*, 2011]. Les événements sont utilisés dans la communauté Computer-Supported Cooperative Work (CSCW) pour la sensibilisation aux activités distribuées [Cai et Yu, 2014] et également l'analyse de conceptions collaboratives [Bang *et al.*, 2017], 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 événements [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 exemples est un tableau Kanban collaboratif temps réel construit sur leur plateforme [13].

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 (EP). Les éditeurs (ceux qui émettent

13. https ://github.com/eventuate-examples/es-kanban-board

des messages) et les abonnés (ceux qui souscrivent) sont couplés de manière lâche. Autrement dit, l'éditeur n'est pas forcément au courant de l'existence de ses abonnés et n'a donc pas de contrainte forte le liant à ces derniers. De son côté, l'abonné est souvent indifférent à l'éditeur 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 l'éditeur et les abonnées. Son rôle est de gérer les abonnements (abonnement / désabonnement) et de notifier les abonnés concernés par la publication d'un événement par un éditeur.

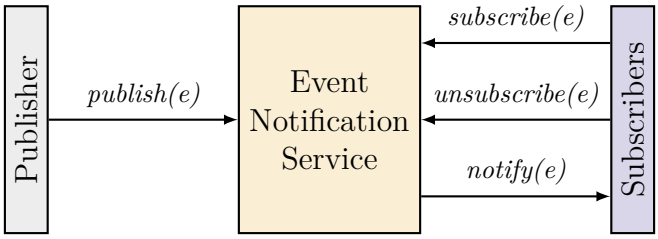


FIGURE 2.5 – Architecture Publish-Subscribe

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

Pour répondre à des besoins de passage à l'échelle, d'interopérabilité, de fiabilité, d'expressivité et d'utilisabilité, un modèle Pub / Sub peut s'appuyer sur :

— le contenu du message transmis (content-based) – l'intérêt de l'abonné est décrit par le type et les valeurs de l'événement qu'il veut recevoir ;

— le sujet du message (topic-based) – le système dédie un canal à des événements d'un type en particulier. Les éditeurs publient sur le canal approprié, alors que les abonnés exprime leur intérêt à recevoir les messages d'un certain type.

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 (*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 orientées sujets. 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 à la grappe (*cluster*) correspondante 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 [Parzyjegla *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 grappe par intérêt (*clustering*) comme moyen 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[Ripeanu,] 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 (client gagnés / clients perdus), résistants aux erreurs, et offrants 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'alors reposent sur une connaissance totale du réseau pour être efficaces, du fait de leur taille limitée. À grande échelle, suivre et collecter l'information de chacun devient complexe. Le modèle Pub / Sub, très peu couplé est une approche efficace pour collecter et analyser les échanges dans les réseaux P2P. La paternité de chaque événement peut être tracée facilement tout au long du routage notamment dans un environnement décentralisé. Il est cependant important de garder en tête que la connaissance peut vite devenir viciée (*staleness*) dans un système dynamique décentralisé, ce qui impose de trouver une méthode efficace pour garantir la fraîcheur des données, leur accessibilité avec une gestion de la dynamicité des pairs efficace.

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 ou 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 propres 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 performance des systèmes collaboratifs utilisant une architecture distribuée. Bien que ces approches permettent

de passer à l'échelle de grandes quantités de données, cela requiert de lourds investissements 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 les utilisateurs (*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 logicielle 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 ambigüités 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.

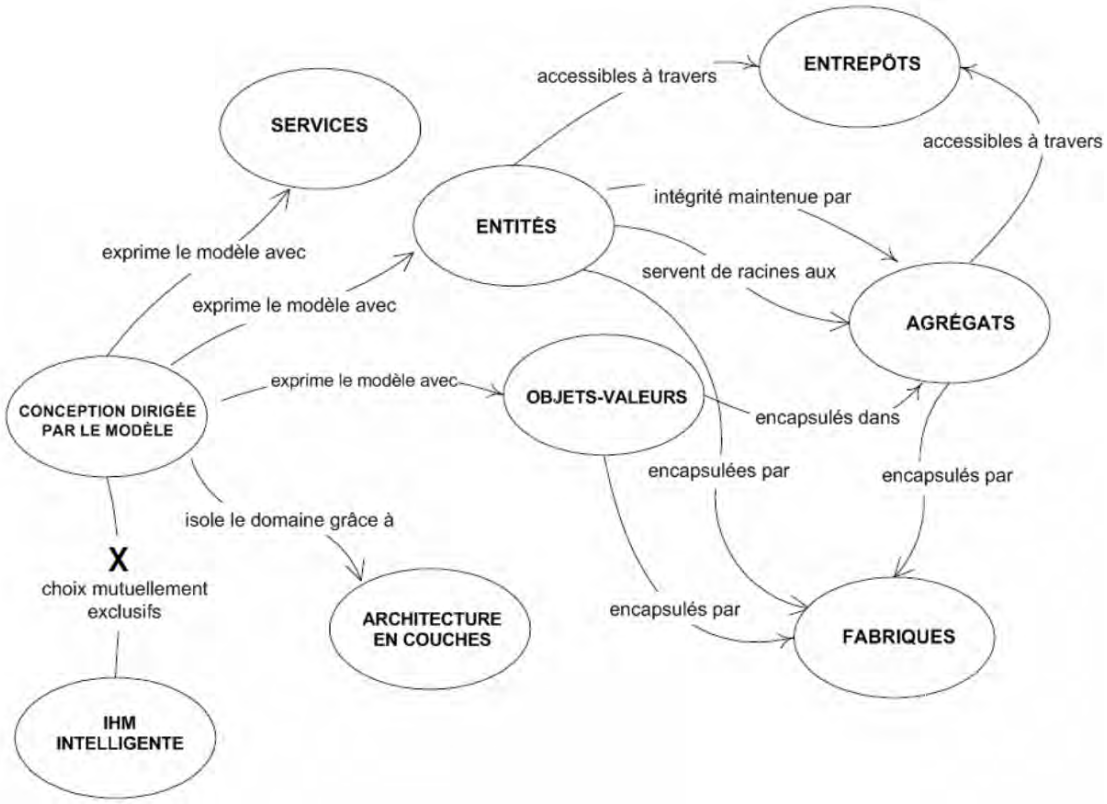


FIGURE 2.6 – Illustration du patron DDD et de ses artefacts (issue de [Avram et Marinescu, 2006] sous licence Creative Commons)

En DDD, il existe des artéfacts 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'intéret 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] [14].

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 dans la

14. 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.

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 Responsability 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 les 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 métiers propres 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.

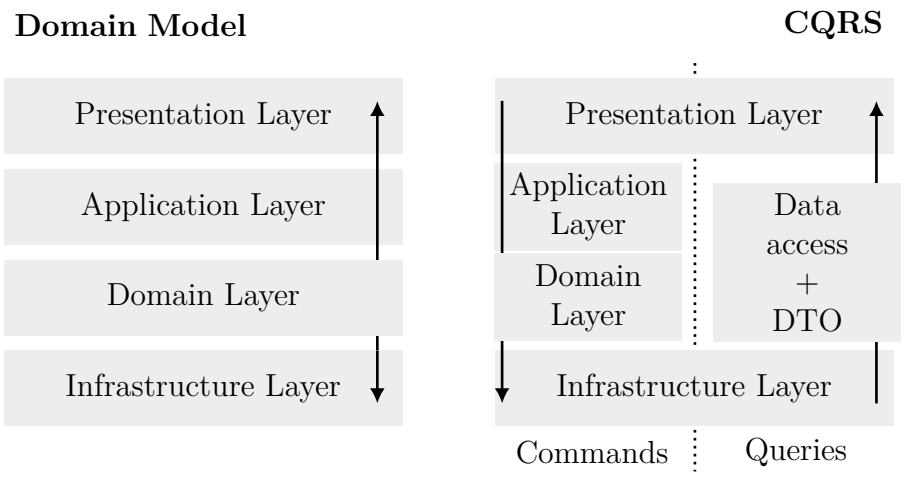


FIGURE 2.7 – Architecture en 4 couches du DDD (gauche) en miroir avec l'architecture CQRS (droite)

Le caractère vicié 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 des 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 actuel. Ce paradigme permet de recréer n'importe quel état d'un agrégat à partir d'une séquence d'événements donnée. Cette séquence représente la source primaire de données du système, i.e. elle immuable et toujours accessible dans le système. L'ES réduit l'essentiel de l'information à l'événement permettant de passer d'un état à l'autre. L'événement est générique et auto-descriptif ce qui permet réduire la complexité des structures de données en encapsulant de la connaissance dans un élément atomique.

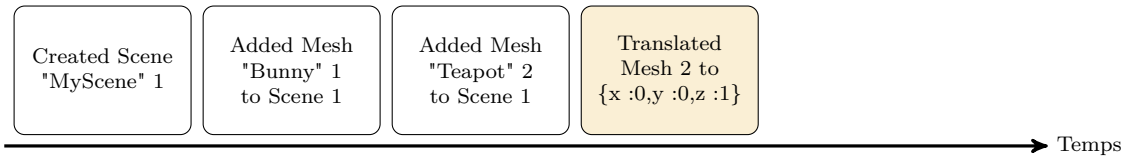
L'ES est souvent présent dans des architecture asynchrones qui ont l'avantage de pouvoir utiliser des queues de message, plusieurs bases de données, et où la partie lecture est éventuellement consistante. La plupart de la littérature concernant ce patron se trouve en ligne, dans des billets de blog, 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

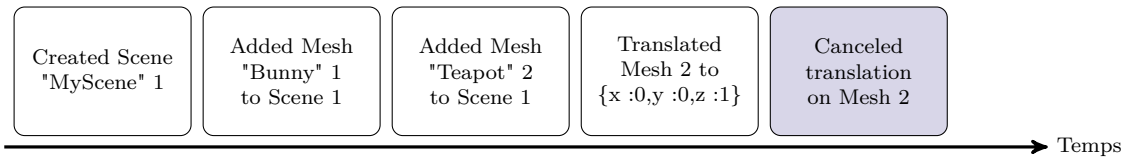
comme immuables et le journal d'événement (*event log*) comme un stockage linéaire (*append only store*) des événements (Figure 2.8a). Les événements ne sont jamais supprimés, le seul moyen de l'« annuler » consiste à générer un événement rétroactif. Une fois qu'un événement rétroactif est ajouté, l'événement 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 particulièrement du 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 défaits. Ce que Fowler appelle événements rétroactifs, Young le décrit comme des actions inverses.

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 domain est persisté comme un *flux* d'événements plutôt qu'un simple instantané ».



(a) Nouvel événement en ES

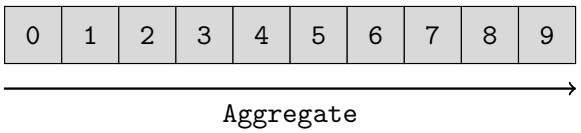


(b) Transaction avec compensation en ES

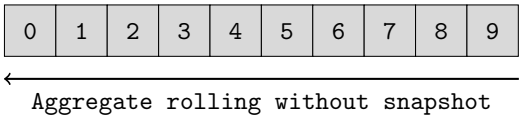
FIGURE 2.8 – Transaction en ES

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.

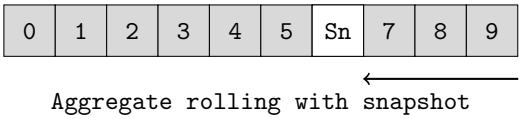
L'ES peut apporter beaucoup d'avantages à une application, notamment lorsque les besoins en traçabilité de l'information sont importants comme dans la modélisation collaborative de données 3D. L'historique du système est accessible tout au long de la vie de



(a) Lecture normale d'un agrégat



(b) Reconstruction sans la notion de Snapshot



(c) Reconstruction avec la notion de Snapshot

FIGURE 2.9 – Snapshot en ES

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 système critiques 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 exprimé par Fowler [Fowler, 2003] 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 : (i) les spécifications ne sont pas assez claires ; (ii) deux collaborateurs sont opposés sur la façon de modifier l'objet. L'origine de ces changements identifiée, le chef de projet peut 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.

Snapshot 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. Le processus va s'alourdir car chaque événement créé (et stocké) ne sera pas supprimé. Plus la pile d'événements est longue, plus cela va prendre du temps (linéaire), surtout en considérant que les commandes ont parfois besoin de connaître une partie de l'état courant issu de ces événements pour être déclenchées (Figure 2.9a).

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. Créer un Snapshot évite de reconstruire un état à partir du début de la vie de l'agrégat (Figure 2.9b) ; les nouveaux événements sont empilés à partir de ce snapshot (Figure 2.9c). La ré-hydratation d'un agrégat est réduite au différentiel qui existe entre le snapshot et l'état courant de la projection. 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 évité 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 : `État → événement → 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 toutes 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) : `État → Commande → 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). Pour les EVCs 3D, et particulièrement pour la modélisation 3D collaborative, 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 nœuds 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

Sommaire

3.1 Introduction 52
3.2 Modèle événementiel pour l'intégration du domaine 3D dans les EVC 53
3.2.1 Modèle général 55
3.2.2 Adaptation des patrons ES et CQRS 58
3.2.3 Journal des événements 60
3.2.4 Flexibilité de la visualisation 63
3.2.5 Cohérence éventuelle en CQRS 63
3.2.6 Bilan 65
3.3 Architecture de communication hybride 66
3.3.1 Éléments constitutifs d'un pair 69
3.3.2 Architecture hybride « orientée états » 72
3.3.3 De l'état à l'événement 76
3.3.4 Architecture de communication hybride « orientée événements » . 77
3.3.5 Bilan comparatif 85
3.4 Conclusion du chapitre 85

3.1 Introduction

Les Environnements Virtuels Collaboratifs 3D (EVC 3Ds) 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 trois dimensions (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'historique 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 (détection 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 des patrons de conception Domain Driven Design (DDD), Command Query Responsability Segregation (CQRS) et Event Sourcing (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 pair à pair (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 présentée 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'EVCs, 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.

Les Architectures Orientées Événements (AOEs) (ou Event Driven Architectures(EDAs) dans la littérature anglophone) sont par nature des systèmes très peu couplés. 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 validées, stockées puis affichées par le système.

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 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 quasi temps 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.

L'EP est devenu un paradigme de choix pour les applications proposant des outils de surveillance et des applications réactives [Hinze *et al.*, 2009]. La compréhension des événements à travers leur composition, leur abstraction, leur type de traitement et la qualité de service offre un large panorama de domaines d'applications et de technologies possibles.

C'est pourquoi il est nécessaire de définir un cadre à la modélisation événementielle : une spécification générique des événements, la définition des événements propres à la modélisation 3D collaborative. La validation des règles métiers doit également être introduite dans le processus de traitement des événements. La visualisation des événements doit être assez flexible pour gérer l'arrivée des flux d'événements produits ou issus de l'intergiciel P2P orienté messages (type Publish-Subscribe (Pub / Sub)) par lequel les pairs se transmettent les notifications d'événement.

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 à 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 (CPU et GPU) 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 (Web Real-Time Communication (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 Plusieurs aspects de la contribution dans le modèle événementiel sont décrites ci-après. Tout d'abord, il est nécessaire de spécifier les événements qui vont être utilisés sous une forme générique. Ces événements, qui représentent ce qui se passe dans l'application, doivent faire ressortir l'expertise de chaque modification. Que ce soit le contexte auquel il est lié, les paramètres qu'il inclut, le type représenté, les informations de chaque événement sont issues du domaine de la modélisation 3D. Puis, le modèle événementiel propose d'adapter les patrons de conception ES et CQRS sous un angle nouveau. En déportant ces patrons en totalité sur le client, le système est capable de traiter le cycle des données de manière autonome (sauf la sauvegarde à long terme) en profitant des ressources que le client met à disposition. La conservation de la séparation du traitement en écriture et en lecture permet au système d'être capable de vérifier la validité des modifications entrantes par lui-même et d'être flexible sur les flux sortants d'événements qu'il propose. Cela simplifie notamment la transmission des données liées à la 3D et à la collaboration en limitant 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 (coupure). L'utilisateur a la garantie d'avoir un historique performant qui profite de chaque connexion au réseau pour mettre à jour le système. Le modèle événementiel présenté dans cette thèse inclut également un modèle de cohérence allégé. Il prend la forme d'un module de détection de conflit basé sur le journal d'événements dont chaque client possède une réplique locale.

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. Dans cette thèse le contexte est l'assemblage d'objets 3D dans une scène 3D. 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 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. Une partie de la contribution est d'adapter les propriétés de cette spécification pour la modélisation 3D collaborative (visualisation et manipulation).

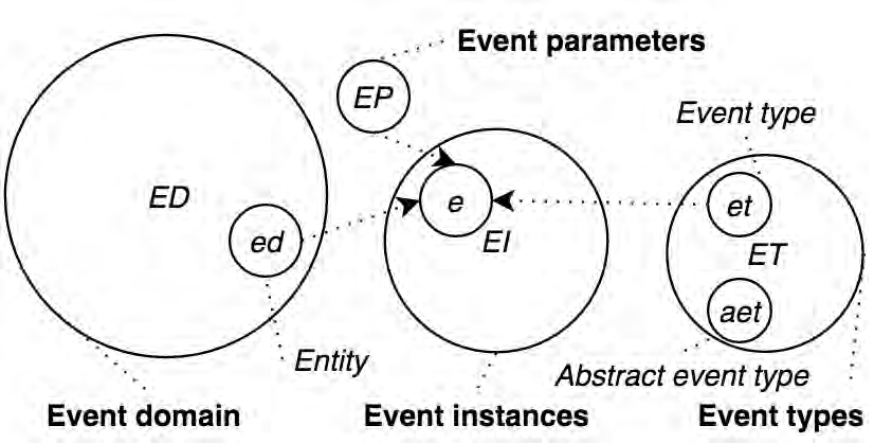


FIGURE 3.1 – Illustration des notions d'événements du domaine, des types d'événements, et des instances d'événements.

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'événement *e* – ou simplement événement – 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 Environnement Virtuel Collaboratif (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 de cette thèse.

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. Pour *meshAdded*, l'utilisateur va, en cliquant sur l'item de la bibliothèque, voir apparaître un maillage au centre de la scène. Concernant *meshDropped*, l'utilisateur effectue un glissé / déposé de l'objet depuis la bibliothèque à l'endroit dans l'environnement 3D où il souhaite que l'objet soit positionné.

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).

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

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. Ces événements qui ne sont alors pas encore sauvegardés dans l'Event Store sont passés en entrée de l'algorithme 3.1. Chaque événement est encapsulés dans un événement « technique » dédié au stockage dans l'Event Store avant d'être publié sur le bus via l'Event Publisher.

Algorithm 3.1 Sauvegarde d'événements d'un agrégat dans l'Event Store

Entrée : aggregateId : string, uncommitedEvents : eventMsg[ ], version : number
  $expectedVersion \leftarrow version$
  for $eventMsg$ **in** $uncommitedEvents$ **do**
    $newMsg \leftarrow \{type : eventMsg.typeEvt, data : eventMsg, streamId : aggregateId\}$
    $publish(aggregateId, newMsg, expectedVersion++)$
  end for

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'Interface Utilisateur (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* ou *active*. La mise à jour passive est utilisée dans le cas des mises à jour liées à la visualisation 3D des modifications des collaborateurs en temps réel par exemple. La projection va envoyer les nouvelles informations de mise à jour sans attendre de la Vue qu'elle les demande. Concernant le cas par exemple d'un changement de scène, les mises à jour sont demandées de manière active et explicite par la Vue à la projection. L'utilisation de ces deux modes dépend du contexte et des besoins de la vue.

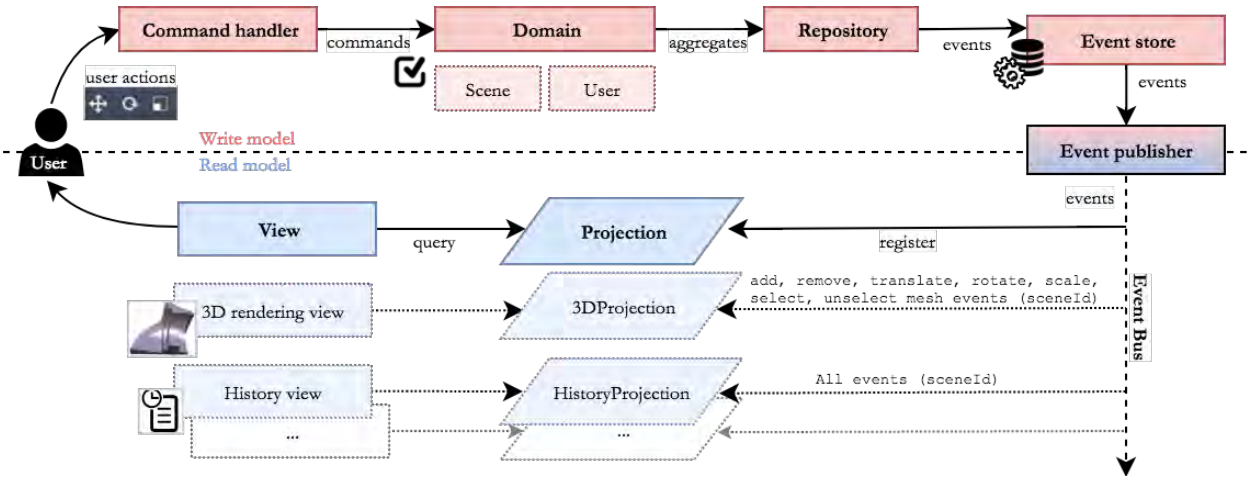


FIGURE 3.2 – Modèle de l'architecture client dans 3DEvent : la gestion du cycle de vie des données avec CQRS et ES dans un navigateur web, des actions des utilisateurs à la visualisation en passant par la synchronisation réseau.

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. Cette validation est nécessaire dans le cadre d'application web où un utilisateur malveillant peut altérer les paramètres en modifiant le code dans le navigateur. Il est également nécessaire de faire cette vérification si le service de gestion des commandes est exposé sous la forme d'une Application Programming Interface (API), pour garantir la sécurité d'exécution des commandes.

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 et du CQRS 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.

3.2.3 Journal des événements

Les événements issus du journal d'événements (ou Event Store), en passant dans la partie « lecture » du modèle CQRS peuvent être consommés par différentes projections accrochées au bus d'événement. De cette manière, une fois le journal d'événements synchronisé, les projections locales sont identiques (à une cohérence éventuelle près). Un journal d'événements est composé d'agrégats versionnés qui contiennent les instances d'événements (Figure 3.3). 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.

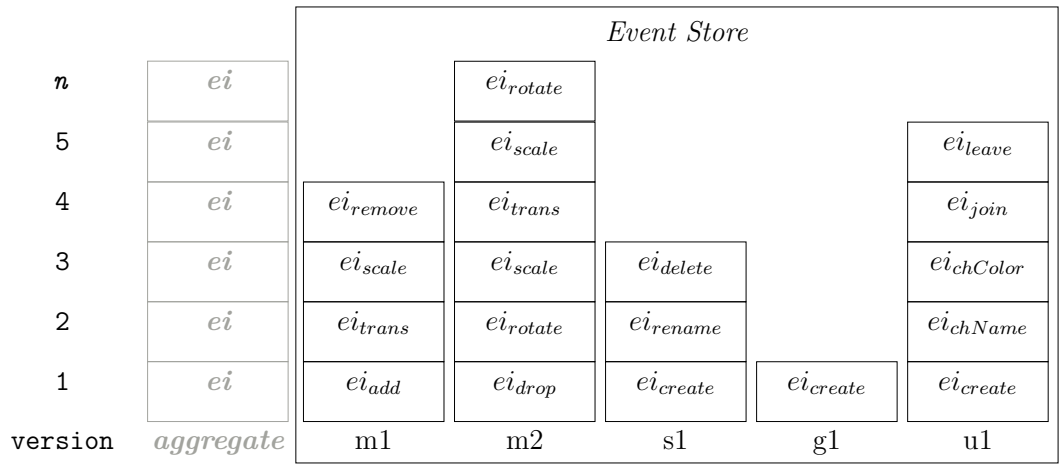


FIGURE 3.3 – Représentation d'un journal d'événement stockant des agrégats (« m1 », « m2 », « s1 », « g1 », « u1 ») versionnés (de 1 à n) qui contiennent des instances d'événements ($ei_{typeEvent}$)

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. Les points d'entrée (aussi appelés Network Bridges (NBs)) 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.

3.2.4 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 persistent, 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 découplée 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.

3.2.5 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

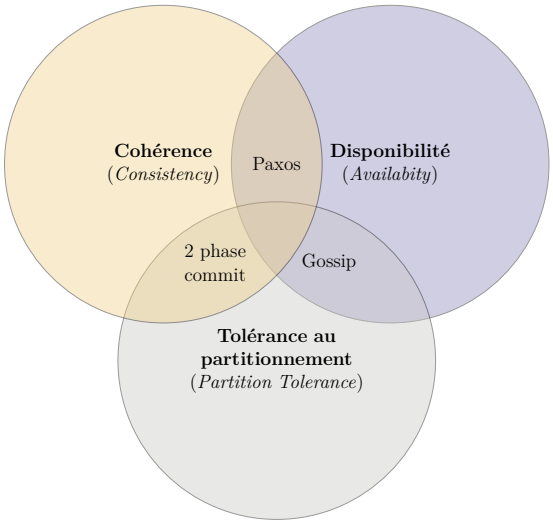


FIGURE 3.4 – Théorème CAP et les algorithmes de compromis

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 évènements respecte l'ordre causal [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 3.4). « *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 ténu 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 découplée 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 de 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'agré-

gat. 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.

L'ES donne la possibilité de gérer les conflits liés à la concurrence avec une fine granularité. Par exemple, selon le scenario suivant : $UserA$ a effectué une une translation $MeshTranslatedEvent$ sur le maillage $mesh1$ au moment où $UserB$ a effectué une rotation $MeshRotatedEvent$ sur ce même objet. Lors de la sauvegarde des événements, la version de l'agrégat $mesh1$ est différente chez chacun des deux utilisateurs. Le module détecteur de conflits contient un registre des procédures à suivre en fonction des événements qui sont en conflit entre eux. L'algorithme 3.2 présente la fonction utilisée pour détecter des conflits dans une liste d'événements donnée. La fonction permet de savoir si un conflit est levé à partir de la liste d'événements donnée et du registre des conflits.

Algorithm 3.2 Registre de conflits

Entrée : $eventToCheck \leftarrow$ list of events to check conflict with
$conflictRegister \leftarrow Dictionary < Type, List < Type >>$
Sortie : return true if a conflict is raised

```
function ConflictsWith(eventToCheck : Type, previousEvents : List < Type >)
    if eventToCheck ∈ conflictRegister then
        for et ∈ previousEvent do
            if conflictRegister[eventToCheck] == et then
                return true
            end if
        end for
    end if
    return false
end function
```

3.2.6 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 3.2.4. 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.

Dans le cas d'application de cette thèse le choix de limiter les événements à des opérations de haut niveau permet de mieux montrer le cloisonnement entre les différents éléments manipulés. En élargissant ce cadre à l'édition de maillage (déplacement de vertex) les problématiques liées à la génération d'événements ne seraient pas touchées. Cependant la gestion de la cohérence lors de la collaboration deviendrait plus complexe avec l'intégration de nouvelles opérations non commutatives. D'autres cas d'usages comme la co-manipulation ne peuvent s'envisager qu'en considérant la possibilité de créer des événements co-générés par les utilisateurs concernés. La synchronisation des deux environnements virtuels serait très différentes de celle considérée actuellement dont la philosophie repose sur la cohérence éventuelle et la mise à jour asynchrone des EVCs.

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 Système d'Edition Collaborative (SEC) adapté aux besoins de l'édition de scènes 3D, plusieurs types de granularité sont à distinguer :

— une granularité liée aux besoins de la collaboration multi-utilisateur ;

— 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 nœuds 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 s'appuient sur une architecture de communication hybride tripartite (Figure 3.5) pour faire de la modélisation 3D collaborative : serveur, persistance et les pairs [1].

Dans un premier temps, l'architecture s'appuie 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. Le verrouillage est géré de manière centralisée par le serveur et la base de données indiquant le statut de l'objet à l'utilisateur le requérant lors qu'il est sélectionné. En cas d'inter-blocage, les objets sont libérés après un certains temps configuré par le serveur. Plus tard, la mise en place du modèle orienté événements impose l'évolution, au niveau réseau, du protocole de transmission (celui-ci reposant sur des notifications d'événements et la gestion des flux d'agrégats). Enfin, la création de deux rôles pour les pairs permet de distinguer leurs responsabilités : 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.

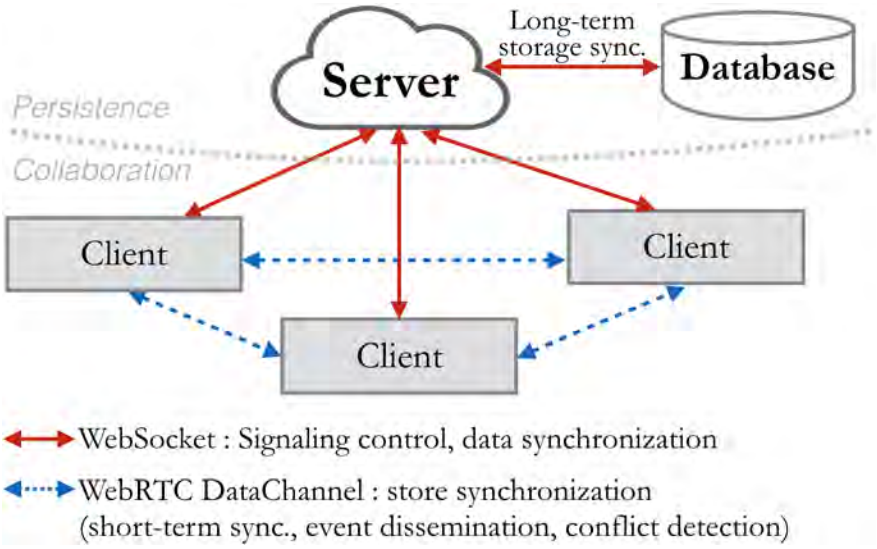


FIGURE 3.5 – Architecture hybride tripartite : serveur, persistance et les pairs.

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

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

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 Commutative Replicated Data Types (CRDTs).

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

Un réseau P2P pour le calcul ou la transmission d'information est un modèle d'architecture qui permet de distribuer les tâches et la charge de travail entre différents pairs. Chaque pair ou « nœud » est à la fois client et serveur, ce qui implique que les pairs sont à la fois fournisseurs et consommateurs de ressources contrairement au traditionnel modèle client-serveur où la production et la consommation sont séparées. Les pairs mettent directement à disposition des autres participants du réseau une partie de leurs ressources (espace de stockage, CPU, GPU) sans nécessiter une coordination centrale par un serveur. En cela, les pairs ont (en général) les mêmes privilèges et participent de façon équipotentielle dans l'application.

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'intergiciel, 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.

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

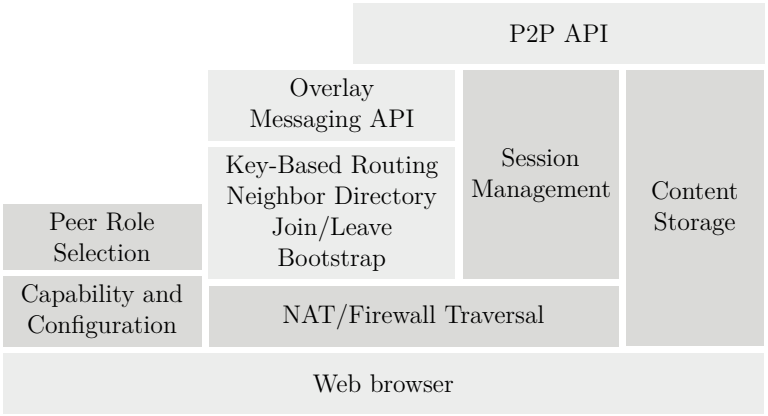


FIGURE 3.6 – Composants de chaque pair dans 3DEvent (point de vue réseau)

P2P. Les étapes pour rejoindre ou quitter la couche sont proposées par le protocole d'établissement de connexion (incluant le mécanisme de *signaling*). 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.

Le contenu partagé est stocké localement sur le pair – 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, selon le modèle, aux différents états des agrégats ou au journal d'événements.

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 dynamicité par exemple. La Figure 3.6 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 :

— *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 les méta information sur les liens entre les collaborateurs. Cela peut permettre au gestionnaire des connexions de mieux évaluer la charge du réseau. Le protocole d'échange correspond à l'interface générique pour que les pairs puissent communiquer entre eux. La définition du protocole est fortement orientée sur la synchronisation.

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 a 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 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 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)) sont transmis par l'usage d'une API REST.

Gestion de la synchronisation et de la cohérence

Le processus de synchronisation utilisé dans cette architecture reste simple dans sa conception et son implémentation. 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. L'échange des mises à jour entre le client (persistance à court terme) et la base de données

(à long terme) leur permet de se synchroniser dans un premier temps. La base de données peut également servir en cas de conflits importants ; elle sert de référence.

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. A la reconnexion les conflits sont gérés manuellement par un utilisateur *master* qui choisit pas-à-pas les opérations à effectuer. Par défaut, le serveur va automatiquement ignorer les opérations produites après un conflit. Seule la détection de conflits est prise en compte, pas leur résolution automatique.
   (c) travail "*online*" (en collaboration entre pairs comprenant la partie serveur) : le client publie systématiquement 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 seul utilisateur est 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 sur les architectures 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. Dans ce modèle, la cohérence est maintenu dès lors que la fréquence des modifications un même objet 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 ($S = \{x_0, x_1, ..., x_n\}$). 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 Sw, 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.

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

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 ?).

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

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 (ex : MeshHisto [Salvati *et al.*, 2015]) 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.

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

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.

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

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]. Contrairement à l'architecture précédemment décrite, celle-ci repose sur le paradigme événementiel et fonctionne en accord avec le framework orienté événements présenté dans la Section 3.2. Trois aspects de l'architecture de communication hybride orientée événements sont mis en valeurs dans cette thèse : la présentation des composants de l'architecture P2P peu couplé, un système de gestion de cohérence ainsi qu'un protocole de synchronisation utilisant un journal d'événements partagé entre les pairs.

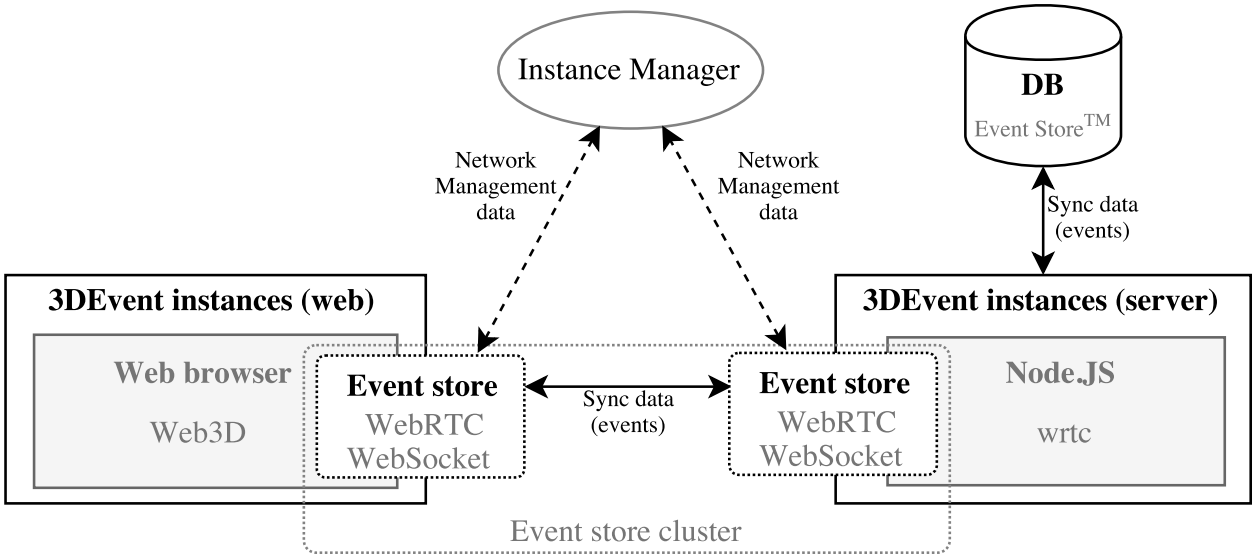


FIGURE 3.7 – Architecture de communication « orientée événements »

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). L'Event Store produit en sortie des événements dits « 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 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 ils sont reliés.

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

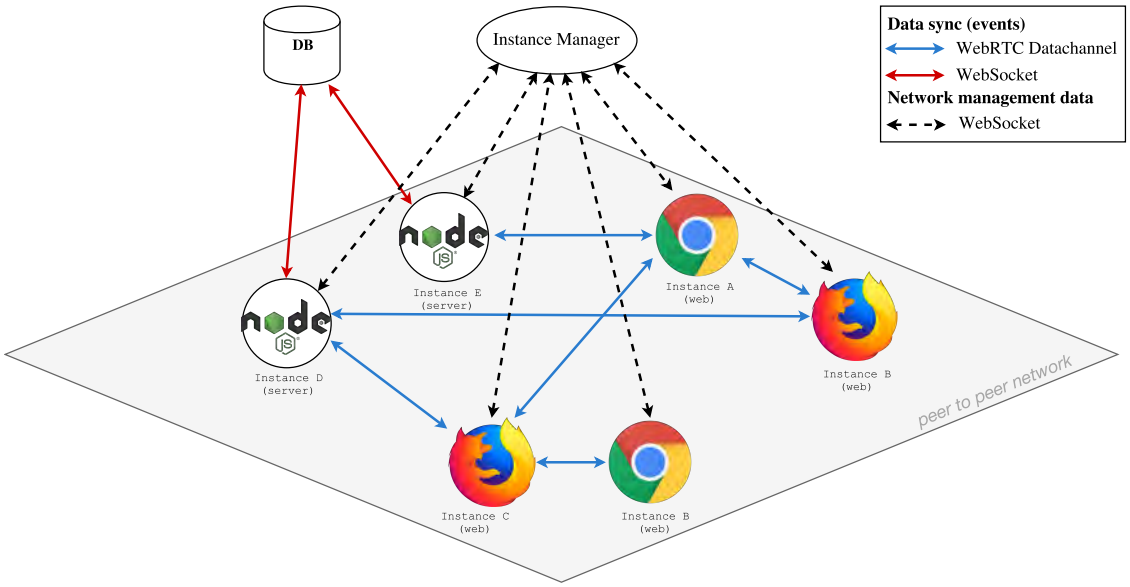


FIGURE 3.8 – Illustration de l'architecture de communication « orientée événements » avec 4 Instances Web et 2 Instances Serveur

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'IM centralise toutes les mises en relation des différentes instances ; la connectivité du maillage est paramétrable à partir de l'IM. Dans le cas où l'IM n'est plus accessible (ex : serveur inaccessible), les différentes instances déjà en relation peuvent continuer à échanger. WebRTC offre également la possibilité de gérer manuellement et sans serveur les connexions entre instances.

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

La Figure 3.8 représente un réseau pair-à-pair possible avec cette architecture. Toutes les instances sont au même niveau dans le réseau pair-à-pair. Chaque nœud contribue à la distribution du contenu au sein du réseau.

Gestion du stockage

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, il stocke l'événement à la suite de ceux présents dans le flux de l'agrégat sur le pair, localement. Par ailleurs, 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 du système stocke le journal

d'événements qui est la source de vérité de l'application. Elle peut également stocker des projections prédéfinies, 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 constituent des données purement fonctionnelles (Section 2.3.5). 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, les données stockées sont immuables – i.e. elles ne peuventt 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.

Gestion de la synchronisation et de la cohérence

Lorsqu'un pair initialise ou rejoint la séquence pour rejoindre une session collaborative, il évolue pour se synchroniser et intégrer les nouveaux événements à son journal. La Figure 3.9 représente la séquence d'actions nécessaire à une instance 3DEvent (*idA*) pour rejoindre le réseau contenant déjà d'autres instances 3DEvent ; elle contient trois phases. La première phase (*peer creation*) consiste à prendre connaissance des autres pairs déjà présents, la seconde phase (*P2P network creation*) permet la création et l'établissement des connexions P2P entre la nouvelle instance et les autres nœuds, puis la troisième (*Real-time collaboration*) correspond au partage en temps-réel des messages entre les différentes instances connectées au réseau P2P.

La première phase est déclenchée par l'action *join*, 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. Cette liste est gérée par le gestionnaire d'instance, qui la retourne au pair pour lui indiquer les pairs avec lesquels il doit se connecter. Pour chaque pair *idB* de la liste retournée *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 méta-données sur la situation respective de leurs ESM afin de se synchroniser.

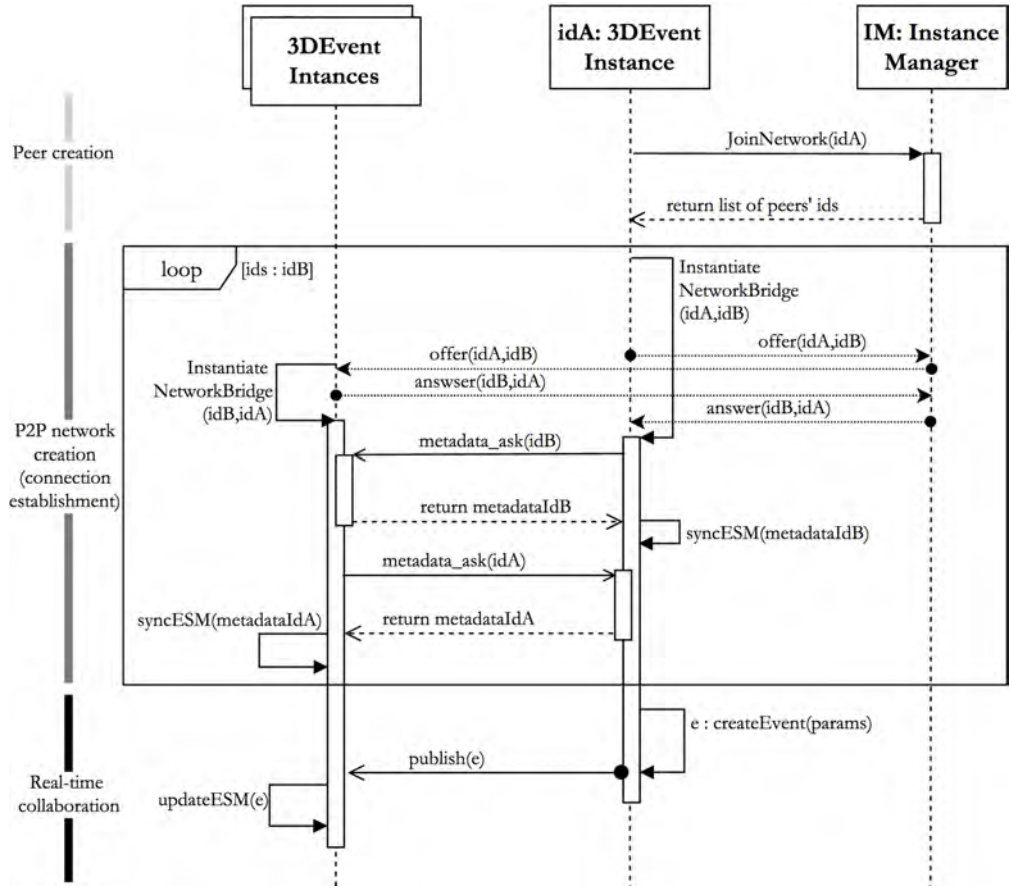


FIGURE 3.9 – Protocole de connexion au réseau d'instance 3DEvent

Cohérence d'un flux d'agrégat 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 identiques, l'événement est ajouté à la fin du tableau du flux pour être stocké dans l'ESM, sinon une exception indiquant l'incohérence est levée.

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. Pour être cohérent, le flux de 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 conflits, on est à même de la comparer avec la version courante v_c. Il existe deux cas de conflits :

a) v_a correspond à la valeur d'initialisation du flux : après une action, la version initiale de l'agrégat ne peut être identique ;

b) v_a est différente de la version v_c : l'Event Store propage une information incohérente si la version attendue est inexacte.

La cohérence est gérée par l'Event Store à plusieurs niveaux dans le pair : localement lors de la génération d'événements par le pair et au niveau du réseau P2P lors de la réception d'événements. Les nouvelles données qui entrent dans un pair doivent nécessairement être en cohérence avec celle qui sont déjà certifiées cohérentes.

L'algorithme 3.3 permet de détecter une incohérence lors de l'arrivée d'un nouvel événement. Cet événement est fourni avec une version qui correspond au numéro de version attendu du flux de l'agrégat (*stream*) auquel il est rattaché. Si le flux n'existe pas, il est créé, sinon il faut vérifier qu'il n'existe pas déjà un événement stocké à la version donnée pour pouvoir ajouter l'événement au flux, sinon une exception indiquant un problème de cohérence est levée.

Algorithm 3.3 Ajout d'un événement dans l'Event Store

Entrée : streamId : string, event : EventStoreEvent, version : number
Sortie : event
  $stream \leftarrow streams.getOrCreate(streamId)$;
  if $stream.has(version)$ **then**
    $throwExceptionVersion(stream, version)$
  end if
  $stream.data.set(version, event)$

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, le réseau a une topologie partiellement maillée (*partial mesh topology*).

Le protocole d'échange est décrit dans l'annexe A.2. En passant de l'instanciation d'une connexion à l'échange de méta-données pour se synchroniser, le pair agit comme une machine à état concernant la transmission des données : c'est la seconde phase (*P2P network creation*) de la Figure 3.9. Les méta-données échangées permettent aux nœuds de savoir « qui possède quoi » sur le réseau pour pouvoir ensuite demander les événements qui leur manquent. Dans un premier temps, l'instance idA créer le Network Bridge associé à l'instance à laquelle elle cherche à se connecter. Une fois la connexion établie, les deux instances échangent l'état de leur méta-données et se synchronisent. Les échangent de méta-données se font de manière asynchrone. Il se peut donc qu'une même information soit demandée à différentes instances. La réponse est ignorée si l'information est déjà présente sur l'instance qui a demandé cette information. Cette mécanique permet d'accélérer la récupération des informations à partir de plusieurs instances.

Algorithm 3.4 Synchronisation d'un nœud 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()
```

Cette étape de synchronisation d'Event Store entre deux nœuds est reprise en détail dans l'algorithme 3.4. Le nœud courant demande le différentiel (*diff*) entre ses flux et ceux du nœud *node* pour connaître quels sont les flux à synchroniser. Si un nœud possède les données demandées il les envoie. Si le nœud courant reçoit à nouveau les mêmes données, elles sont ignorées par le système.

Discussion sur l'architecture de communication « orientée événements »

L'architecture de communication orientée événement utilise un intergiciel réactif : elle consomme des notifications d'événements et permet le traitement asynchrone de messages. Bien que le découplage proposé par ce système offre des avantages importants dans un fonctionnement décentralisé, il peut également être à l'origine d'incertitudes causées par les délais des notifications, notamment lors de l'apparition de conflits.

Tous les pairs qui participent à la gestion de données (producteurs et consommateur) sont au même niveau dans la couche P2P. Cette situation ne propose qu'une flexibilité limitée dans le passage à l'échelle et dans la gestion de l'attrition. L'équilibre entre les producteurs et les consommateurs doit être assez stable pour ne pas surcharger certaines parties du réseau. C'est une des raisons pour lesquelles le réseau P2P est renforcé par la présence de pairs destinés à faire le relais dans la distribution des données.

CRDTs basés opérations La gestion de la cohérence dans un SEC massif est souvent étudiée à travers le prisme des Commutative Replicated Data Types (CRDTs). Les CRDTs sont des types de donnés répliqués qui convergent prenant en compte des mises à jours concurrentes. Un CRDT peut être mis à jour sans nécessiter la coordination des répliques. Cela rend les CRDTs hautement disponibles pour les opérations d'écriture. Les CRDTs peuvent être classés selon deux catégories : les CRDTs orientés état (CvRDTs ou ***convergent*** *replicated data types*) et les CRDTs basés opération (CmRDTs ou ***commutative*** *replicated data types*). Les CRDTs orientés état sont conçus pour disséminer un état parmi les répliques alors que les CRDTs basés opérations sont conçus pour disséminer des opérations. Les répliques de CmRDT sont garanties de converger si les opérations sont disséminées au travers d'un intergiciel à transmission fiable et causale (*reliable causal broadcast* (RCB)) et si leurs opérations concurrentes sont commutatives. Les CvRDTs ne nécessitent pas d'intergiciel de messagerie mais une bande passante croissante dû à l'augmentation de la taille de l'état. La convergence passe par une fonction de fusion définie comme une jointure.

L'exécution d'une opération sur un CmRDT s'effectue en deux phases, **préparation** et **effet** (aussi appelées *atSource* et *downstream* [Shapiro *et al.*, 2011]. La production du message représentant l'opération se base sur l'opération et éventuellement l'état courant pendant la phase de préparation du CmRDT.

Relation entre CRDTs et Event Sourcing Les deux phases de mise à jour d'un CmRDT, **préparation** et **effet**, sont très proches des phases de mise à jour des entités en ES, **traitement de la commande** (*command handler*) et **traitement des événements** (*event publisher*). Durant le **traitement de la commande**, une commande entrante peut

être validée en fonction de l'état de l'entité sur lequel elle est appliquée et, si elle l'est, un événement représentant l'effet de la commande est inscrit dans le journal d'événements. Cela correspond à la phase de préparation d'un CmRDT. Durant la phase de **traitement des événements**, l'événement écrit est consommé à partir du journal d'événements et utilisé pour mettre à jour l'état courant de l'entité. Cela correspond à appliquer la représentation de l'opération produite à l'état du CmRDT local durant la phase **effet**. Le CQRS associé à l'ES distingue ces deux phases et fournit une couche d'abstraction supplémentaire permettant à l'application de définir des commandes et des traitements d'événements spécifiques au domaine. Les CmRDTs et les entités issus de l'ES doivent pouvoir à la fois consommer des événements issues d'autres acteurs et produire des événements pour les autres acteurs. Cette collaboration est nécessaire pour déployer plusieurs répliques de la même entité.

La représentation des données 3D dans cette thèse ne permet pas l'utilisation de CmRDTs comme dans ChainVoxel [Imae et Hayashibara, 2016]. Cette représentation combinée à notre modèle CQRS et ES pourrait permettre d'améliorer le passage à l'échelle et garantir la convergence des répliques de manière formelle.

3.3.5 Bilan comparatif

Le Tableau 3.2 propose un récapitulatif des deux approches utilisées. HybridEvent correspond à l'architecture hybride « orientée événements » tandis qu'HybridState est l'architecture « orientée états ». Il est important de noter qu'HybridEvent a été conçue après HybridState et que les choix de modélisation ont évolué avec le temps. Globalement, HybridEvent a une approche très peu couplée par rapport à HybridState que ce soit concernant la connectivité, le type de transmission ou encore le mode de cohérence choisi.

HybridEvent propose une gestion des données mieux répartie compte tenu de la possibilité de « composer » le réseau P2P en fonction des besoins avec l'ajout à la volée de nouvelles instances Serveur. Cependant, le choix d'une synchronisation par les pairs rend la garantie de la cohérence plus complexe à faire aboutir contrairement à HybridState. Cette dernière consomme les ressources par à-coups alors qu'HybridEvent repose principalement sur la transmission de flux de données en continu.

3.4 Conclusion du chapitre

Dans ce chapitre est décrit l'ensemble des contributions théoriques de la thèse. Dans un premier temps, le modèle événementiel propose une architecture complète permettant à un client d'intégrer les règles métiers liées à la manipulation et la visualisation d'objets

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	Flexible	Verrouillage des objets
Robustesse	Plusieurs pairs sont liés au serveur BDD : disponibilité ++	Lourde charge pour le 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)

3D. Le modèle événementiel est utilisé pour traiter les événements à plusieurs niveaux : validation des données en fonction des règles métier, du stockage, de la détection de conflits interne et de la visualisation flexible. Tous ces aspects permettent d'ouvrir sur la deuxième contribution : l'architecture de communication hybride combinant client-serveur et P2P. Cette seconde partie se partage entre une architecture « orientée états » et une architecture « orientée événements ». Pour chacune de ces architectures, chaque composant est décrit en fonction des spécificités du paradigme utilisé.

L'utilisation de la technologie WebRTC permet de proposer une architecture qui respecte les standards du web et l'interopérabilité nécessaire aux EVCs 3D. 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 sur le réseau. Cela permet à la fois de récupérer du contenu rapidement, et d'octroyer une autonomie aux contributeurs.

Différentes problématiques liées à la cohérence et la synchronisation ont été évoquées. L'architecture semi-décentralisée proposée fournit plusieurs pistes pour la détection de conflits. Les choix effectués lors de l'implantation permettront d'analyser les différents échanges entre tous les composants de chaque modèle et de souligner les avantages et les inconvénients des choix de modélisation exprimés dans les contributions.

Chapitre 4

Prototypes d'application web pour la conception 3D collaborative

Sommaire

4.1 Introduction 88
4.2 3DState : une preuve de concept orientée états 90
4.2.1 Implantation des composants pour la communication 90
4.2.2 Interface de 3DState 91
4.2.3 Gestion de la session 91
4.2.4 Bilan critique 92
4.3 3DEvent : une approche découplée orientée événements 94
4.3.1 Intergiciel pour l'échange de 3D par événements 94
4.3.2 Interface orientée tâches 99
4.3.3 Sélection fantôme 101
4.3.4 Bilan 103
4.4 Conclusion du chapitre 105

4.1 Introduction

Ce chapitre présente deux prototypes, chacun implanté sur un paradigme différent. Le prototype 3DState repose sur des structure de données orientées états pour proposer une preuve de concept concernant principalement l'architecture de communication hybride, tandis que le prototype 3DEvent s'établit sur le paradigme événementiel pour implanter une approche peu couplée dans l'intergiciel P2P et l'interface orientée tâche par laquelle l'utilisateur interagit avec le système. Chaque prototype possède les mêmes blocs : une interface utilisateur (contenant un environnement 3D) et un intergiciel P2P.

La programmation réactive (PR) est le paradigme de programmation à la base de l'implantation des interfaces graphiques des deux prototypes : les composants répondent à des événements qui leur parviennent. Cela s'applique principalement à l'interface graphique mais également au niveau des flux réseaux pour proposer une application disponible (répondre rapidement), résiliente (disponible même en cas d'erreur), souple (fonctionnelle même surchargée), orientée messages (asynchrones). Pour conserver ces propriétés, la programmation réactive impose de suivre la variation des valeurs dans le temps, être à l'écoute des événements qui surviennent, suivre les dépendances des variables pour pouvoir propager les mises à jour de valeurs, et enfin propager automatiquement les changements. En cela, le langage de programmation JavaScript (JS) s'adapte bien à la PR car il repose sur un système de boucle d'événements, un gestionnaire d'événement et d'injection de comportements asynchrones.

Signaling L'assomption est faite que tous les clients utilisent des navigateurs qui implémentent et supportent le protocole WebSocket et l'API RTCDataChannel de WebRTC. Pour rappel, 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), ordonné ou non ordonné (Section 2.2). La connexion entre un pair (client) et le serveur est établie sur la base du protocole WebSocket. Cette connexion bidirectionnelle 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.

Interactions de base 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 multiplateforme. 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 (Graphic Processing Unit (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.

Dans chacun des prototypes, les interactions de bases sont les suivantes :

— *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 les formats de fichier 3D les plus populaires (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.

Ces interactions, communes aux deux prototypes présentés, ne représentent cependant pas l'entièreté des fonctionnalités nécessaires à un environnement pour la manipulation d'objets 3D de manière collaborative. Par exemple, les fonctionnalités d'historique et de sensibilisation à la collaboration sont plus développées dans 3DEvent que dans 3DState.

Les travaux présentés s'inscrivent dans un contexte où les besoins d'interopérabilité et de standardisation sont élevés pour permettre à des utilisateurs de prendre le système rapidement en main sans installer autre chose qu'un navigateur.

4.2 3DState : une preuve de concept orientée états

3DState correspond au prototype de issu de l'architecture orientée états présenté dans la Section 3.3.2. Il présente l'implantation de l'architecture de communication orientée états ainsi que les choix techniques qui l'accompagne.

4.2.1 Implantation des composants pour la communication

Serveur Le prototype utilise un serveur Node.js qui sert à la fois de serveur de *signaling* pour mettre en liaison les utilisateurs mais il sert également de lien avec la base de données. Node.js permet d'utiliser du JS côté serveur, simplifiant la compréhension et la maintenance de l'environnement. L'application utilise le micro-framework ExpressJS par dessus Node.js qui a l'avantage de simplifier le routage réseau.

Base de données Pour assurer la sauvegarde au long-terme des modifications faites par les utilisateurs, la base de données MongoDB stocke l'état de chaque scène de l'application. L'utilisation d'une base de données NoSQL orientée document permet de stocker toutes les données pertinentes ensemble, dans un même document. Un document est auto-descriptif et peut imbriquer des données dans une structure d'arbre hiérarchisé. Une collection regroupe des documents. Dans ce prototype, il existe trois collections : *Scenes*, *Geometries* et *Meshes*. La collection *Scenes* contient chaque document concernant une scène : identifiant de la scène, métadonnées de l'espace virtuel (nom de la scène, liste des utilisateurs), liste des maillages inclus (identifiant maillage). La collection *Geometries* comprend les géométries importées dans l'application sous la forme de : identifiant de la géométrie, données 3D. La collection *Meshes* représente les maillages utilisés dans les scènes : identifiant du maillage, nom du maillage, identifiant de la géométrie. Les paramètres liés aux opérations CRUD de la base de données sont fournis par la requête REST.

Couche P2P La compatibilité entre navigateurs (*cross-platform*) concernant l'API Data-channel issue du standard WebRTC n'est pas encore effective. C'est pourquoi ce prototype ne fonctionne que sur les dernières version de Chrome (v49+), Firefox (v55+) et Opera (v47+). PeerJS [1] est une bibliothèque *open source* qui enveloppe WebRTC et fournit une API de connexion navigateur-à-navigateur. Le client possède un identifiant, donné par le serveur de *signaling* lors de sa première connexion, qui lui permet de se connecter à un pair distant dont il a obtenu l'identifiant par le serveur. Le mécanisme de *signaling* est délégué à PeerJS

1. PeerJS http://peerjs.com

qui se charge de l'implantation du ICE Framework et des problématiques Network Address Translator (NAT).

Couche applicative Les différentes actions utilisateurs sont relayés par un système événementiel. L'interface intègre une couche de messagerie pour notifier les composant d'un nouvel événement JS. Dans le prototype, la bibliothèque *js-signals* sert de gestionnaire d'événements aux différents composants de l'IU pour communiquer. Dans ce contexte chaque *signal* possède un contrôleur, qui simplifie le contrôle de la réaction des événements en fournissant des fonctionnalités de souscription et de diffusion aux événements JS.

En comparaison avec une implémentation *Event emitter / dispatcher* et Pub / Sub, le patron de conception Observateur dans *js-signals* n'utilise pas de chaîne de caractère pour décrire les types d'événements JS (évite les erreurs de frappe). Une instance *signal* enregistre des procédures (événements JS, callbacks) qui peuvent être asynchrones. Lorsqu'il est intercepté – n'importe où dans le contexte de l'application – les procédures qui lui sont associées sont également déclenchées.

4.2.2 Interface de 3DState

L'implantation de l'interface a une forte dépendance avec les contraintes liées au rendu 3D dans le navigateur. Le framework ThreeJS a été choisi pour utiliser WebGL qui utilise le paradigme impératif, largement adopté dans la communauté du web 3D.

La Figure 4.1 représente l'interface de l'éditeur implantée. L'utilisateur a accès à une liste des scènes. Lorsqu'une scène est sélectionnée elle est présentée ainsi : titre de la scène, nom des collaborateurs présents, outils d'édition, environnement 3D, détails de la scène et informations liées au client.

Dans ce prototype, présentant une preuve de concept pour l'architecture de communication orientée état, les outils d'édition sont représentés de manière rudimentaire. Les trois actions possibles sur un objet sélectionné sont représentées par les boutons « translate » (translation), « rotate » (rotation) et « scale » (homothétie). Les actions peuvent être faites dans le repère « local » ou dans le repère « monde » selon l'état de la case à cocher « local ». L'utilisateur à peu de retours indiquant le résultat de ses actions.

4.2.3 Gestion de la session

Lorsqu'un utilisateur rejoint une session, il suit le déroulement des opérations décrit dans la Figure 4.2. 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

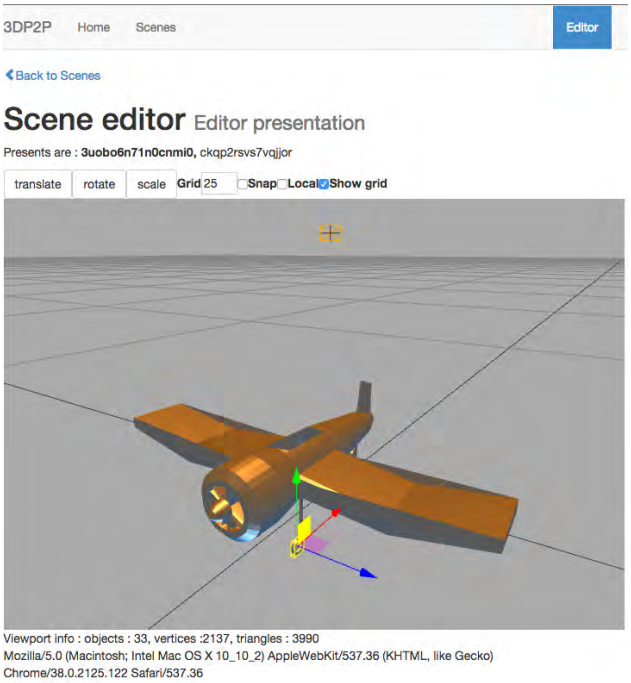


FIGURE 4.1 – Interface de 3DState

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 notifie 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.

4.2.4 Bilan critique

Le prototype 3DState correspond à l'implantation du modèle orienté états appliqué à la modélisation 3D collaborative sur le web. Ses principales fonctionnalités sont : la mise en relation des clients, l'utilisation d'une interface web pour visualiser / ajouter / modifier des modèles dans une scène 3D. Chaque scène peut accueillir plusieurs collaborateurs et permettre à chacun de voir les modifications en temps réel des autres. La couche applicative

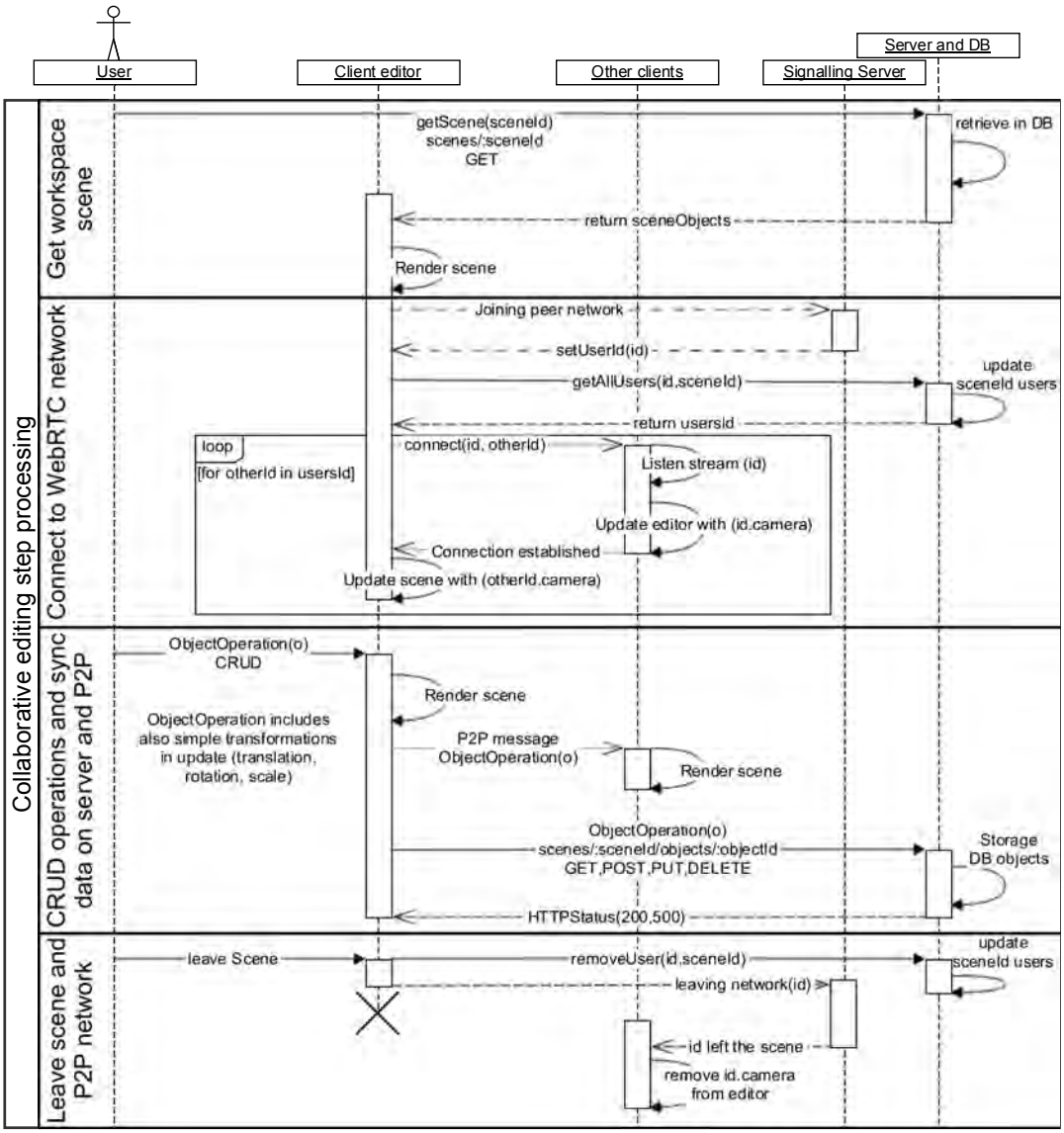


FIGURE 4.2 – Diagramme de séquence de la gestion de la session dans une architecture « orientée état »

utilise la programmation événementielle pour que les composants de l'application réagissent directement aux modifications des objets dans l'environnement 3D.

L'implantation naïve que propose ce prototype pose plusieurs problèmes. D'une part, l'interface étant restreinte au minimum, il manque plusieurs fonctionnalités fondamentales pour que la modélisation collaborative s'effectue dans un environnement adéquat. Par exemple, il n'y a pas de liste des objets présents dans une scène et l'historique de la scène est également absent. La sensibilisation à la collaboration est quasiment absente (couleur différente par collaborateur) à l'exception de la représentation des collaborateurs par leur caméra dans l'environnement 3D. Stocker les éléments 3D par leur état impose un mise en mémoire coû-

teuse et des transferts d'autant plus coûteux par RTCDatachannel. Le Tableau 4.1 résume les différents choix techniques effectués dans 3DState.

Tableau 4.1 – Résumé des choix techniques pour 3DState

	Plateforme / Service	**Bibliothèque (version)**
Pair	Rendu WebGL	Three.JS (r69)
	Gestionnaire d'événements	signal-js (v1.0.0)
	WebRTC	PeerJS (v0.3.9)
Serveur	Node.JS (v0.10.32)	ExpressJS (v4.9.0)
	WebSocket	PeerJSServer (N/A)
	Base de données NoSQL	MongoDB (v2.6.8)

Testabilité Les composants de l'application sont fortement liés entre eux, ce qui rend la testabilité de l'application difficile. La testabilité est réduite au test manuel de l'interface car les différents niveaux d'interaction ne sont pas disjoints (local, distant, collaboratif). D'un point de vue réseau, cette architecture ne permet pas de mesurer de manière objective des temps de réponse et d'échange du fait de cette implantation. Cependant, WebRTC intègre une API permettant de récolter des statistiques sur les échanges qui auraient pu être collectée afin d'observer les échanges durant une session collaborative. Ce faisant, l'observation de la collaboration reste compliquée dans un environnement implanté comme 3DState car l'approche orientée états ne rend pas compte des intentions des utilisateurs.

4.3 3DEvent : une approche découplée orientée événements

Dans le chapitre précédent, une des contributions présentées correspondait 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.

4.3.1 Intergiciel pour l'échange de 3D par événements

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 nécessaires à l'échange de données dans l'application.

Signaling et connectivité

L'assomption 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 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'identifiant 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).

C'est le gestionnaire des instances qui gère la politique de connexion des données lors de la phase de *signaling*. 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 ne 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.

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.

Données d'échanges

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 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 utilisée par le client 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 maintenue grâce à ce numéro de version. Les événements qui gèrent les géométries (*importGeometryEvent*) ont une propriété concernant les données 3D 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

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 une plus grande autonomie des utilisateurs 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. 4.3.1) sans passer par le serveur.

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], 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 la manière dont le contenu est généré : en direct (*live* ou *push*) ou à la demande (*on-demand* ou *pull*).

Les requêtes sur le contenu que les nœuds doivent récupérer sont effectuées selon un ordre de priorité déterminé par l'intérêt que les nœuds portent au contenu. Par exemple, les scènes sont récupérées en priorité par rapport aux maillages et aux informations des autres utilisateurs.

Les données sont transmises sous forme de morceaux (*chunk*) de 16 octets maximum car le navigateur Chrome impose une limite sur la taille des messages envoyés. L'application se charge donc de la création des morceaux et de les envoyer. Ceci a été intégré dans le protocole d'échange de l'application. C'est une des raisons qui a encouragé le choix d'un transport non fiable et non ordonné car l'application est déjà en charge de manipuler les données qui vont transiter sur le réseau. D'autre part, le standard WebRTC n'étant pas implanté de la même manière sur tous les navigateurs, le fait de déléguer une partie de la gestion des données à l'application permet de mieux gérer le traitement qu'elles subissent. Les effets collatéraux

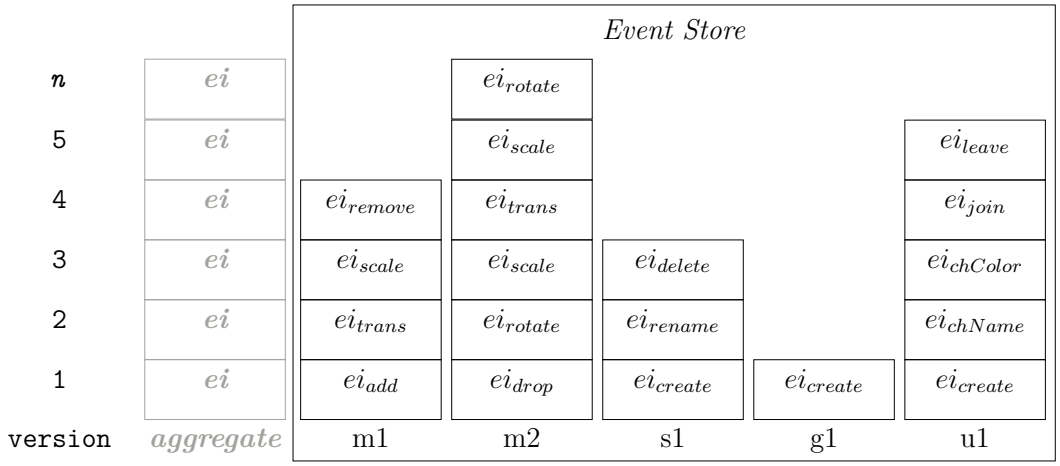


FIGURE 4.3 – Exemple d'agrégats
Structure d'un agrégat et ses versions. Chaque version est un état de l'agrégat qui correspond à l'empilement des instances d'événements (ei) qu'il contient. Les types des événements sont relatifs au type d'agrégat dans lequel il est contenu.

à ce choix sont doubles : la taille des messages qui transitent est allégée, ce qui nécessite d'envoyer plus de messages pour maintenir la la cohérence et la synchronisation.

Gestion des événements dans le flux d'un agrégat

Une connexion RTCDataChannel peut être configurée selon les critères de fiabilité et d'ordonnancement. Dans l'intergiciel 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 pour l'application de les ordonner lorsqu'elle les réceptionne. Lors de la synchronisation des pairs, 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 flux de l'agrégat. 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 2.3.5) qui réduisent la taille des flux en mémoire pour chaque agrégat qui part déjà d'un état avancé.

4.3.2 Interface orientée tâches

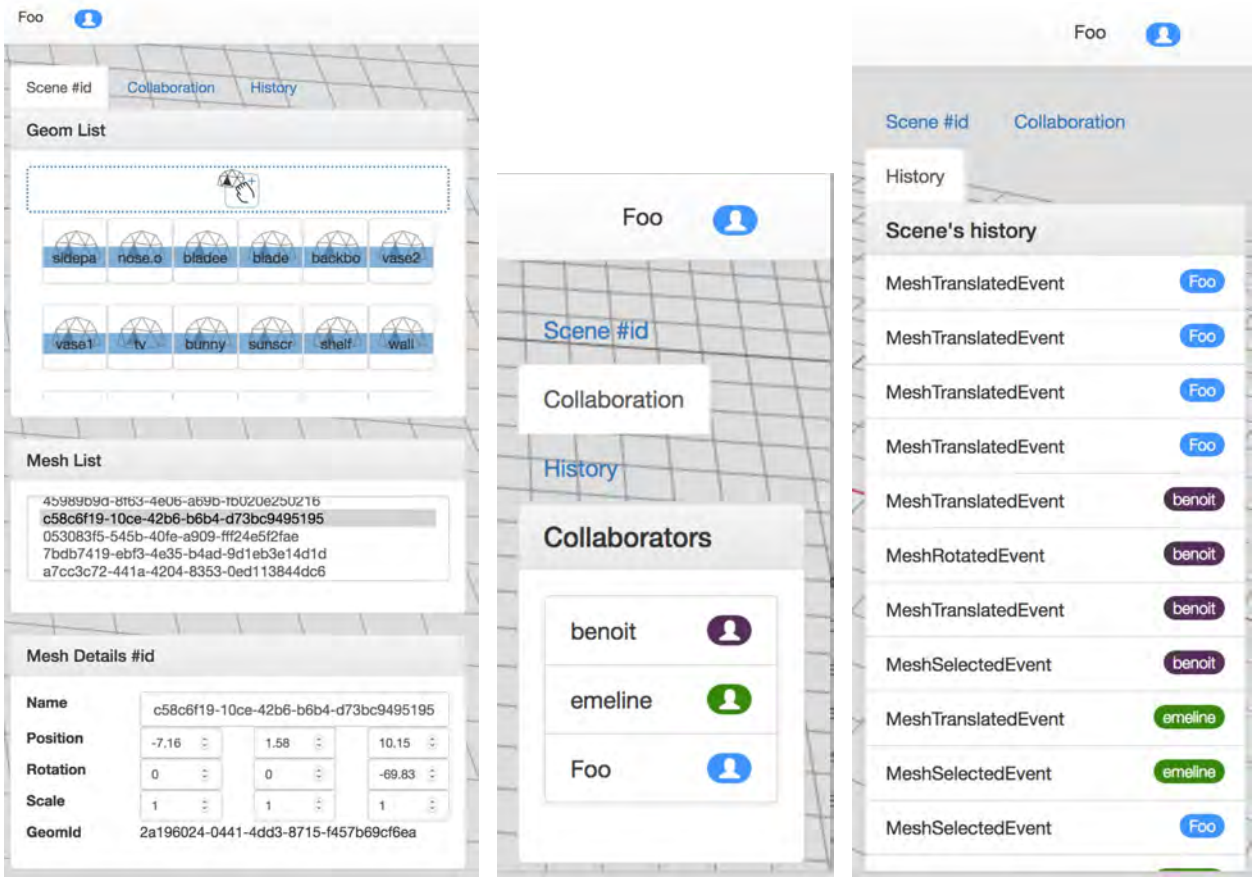
Dans le but de proposer une IU proche des fonctionnalités métiers liées à la modélisation 3D, l'éditeur possède une interface orientée « tâche », en comparaison avec des IU CRUD. En effet, les IU CRUD réduisent la sémantique métier du domaine d'application à la création, la lecture, la mise à jour et 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, qui correspond à la qualité de l'expérience utilisateur fournie 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'agréger, 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 permet 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 eut lieu dans la scène et leurs détails.

La Figure 4.4 montre quelques captures d'écran durant une session collaborative sur le modèle Rotor. L'onglet "Scene" (Figure 4.4a) possède un bloc contenant les détails d'un



(a) Onglet « outils de manipulation sur la scène »

(b) Onglet « collaboration »

(c) Onglet « historique »

FIGURE 4.4 – Onglets du panneau latéral de l'interface

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 4.4b) 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 autre [2]). En cliquant sur un élément de la liste, l'utilisateur accède au dernier point de vue dans l'espace 3D connu du collaborateur représenté.

L'onglet "History" (Figure 4.4c) 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

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

perdre les données entre $state_n$ et $state_x$ car dans le système présenté ici, cela s'effectue par compensation (rappel cf Figure 2.8b).

Dans chaque onglet se trouvent différents blocs HyperText Markup Langage (HTML), avec des comportements spécifiques à un agrégat et injectés dynamiquement. Ces blocs correspondent à des parties de l'interface (vues) dans le prototype.

Parmi les vues 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 vues sont disponibles pour un autre type d'utilisation destinée à l'observation des comportements des utilisateurs, élément primordiale dans cette expérimentation.

Exemple d'interaction

La Figure 4.5 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 collaborateurs [3]. Dans l'étape (a), la commande déclenchée par l'utilisateur s'adresse à l'agrégat *cube*1 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 module détecteur de conflits permet au développeur d'implémenter ses propres règles de résolution de conflit (voir Section 3.2.5). Le module déclenche une exception lorsque le numéro de version reçu et le numéro de version courant de l'agrégat sont identiques (Figure 4.5 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.

4.3.3 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 Wins* (le dernier gagne). La Figure 4.6 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

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

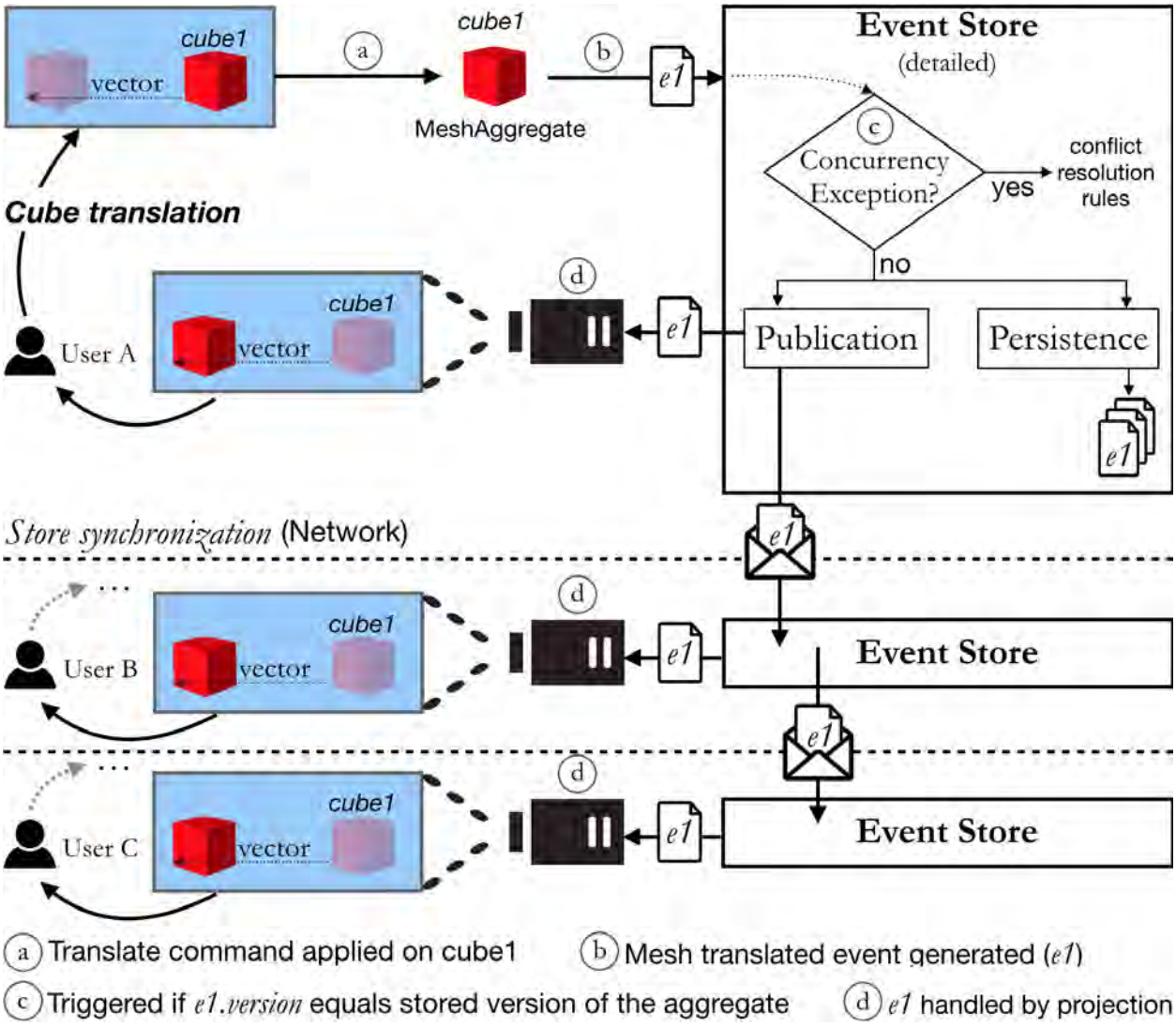


FIGURE 4.5 – Exemple d'édition collaborative où User A est connecté à User B, lui même connecté à User C. Le cycle montre les différentes étapes du déclenchement : la commande, la génération de l'événement, la synchronisation du journal d'événements, l'impact sur le rendu des autres utilisateurs pour une translation sur un cube et le rendu visuel.

représente sa position originale. En différenciant l'actuel objet que l'utilisateur souhaite sélectionner (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 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 transiter est l'événement correspondant à la modification métier et non 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.

4.3.4 Bilan

L'application 3DEvent repose sur les principes et les technologies du web et permet de visualiser et manipuler des objets 3D de manière collaborative en temps-réel. Son approche orientée événement est très peu couplée. L'interface orientée tâches est un bon exemple car elle permet de mettre à jour facilement les blocs concernés. L'interface optimise les performances de rendu en limitant les rafraîchissements aux seuls endroits utiles à la réalisation

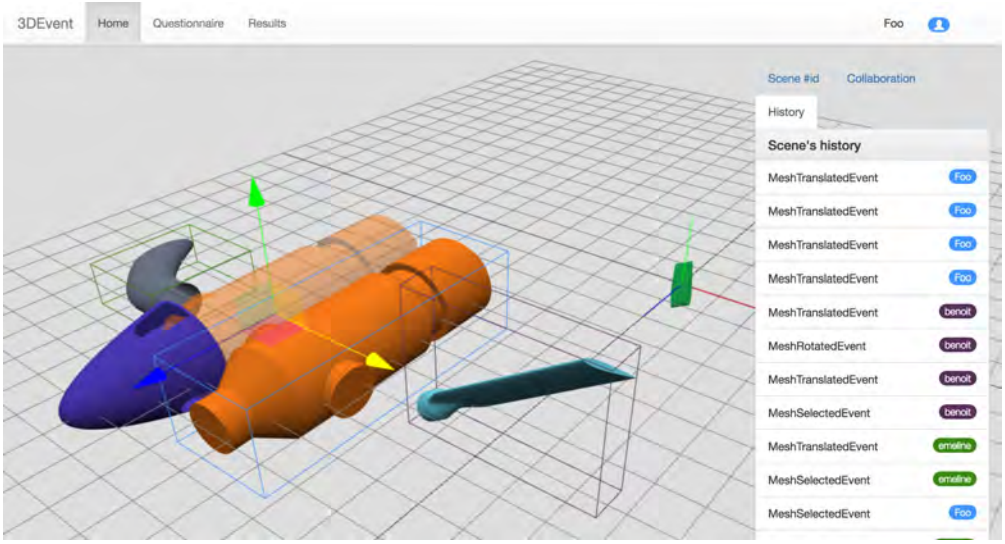


FIGURE 4.6 – Illustration de la sélection fantôme dans l'environnement 3D
Les boîtes englobantes représentent la sélection des différents collaborateurs pendant la session.

Tableau 4.2 – Récapitulatif des choix techniques de 3DEvent

	Plateforme / Service	**Bibliothèque (version)**	**Développements**
Instance Web	Rendu WebGL Gestionnaire d'événement WebRTC	Three.JS (v0.84) signal-js (v1.0.0) peerjs	 CQRSBus webrtc-eventstore
Instance Serveur	WebRTC NodeJS (v8.7)	node-webrtc ExpressJS (v5.0.0)	webrtc-eventstore
BDD		EventStore (v4.0.3)	

de la tâche. Le tableau 4.2 propose un récapitulatif des choix techniques effectués pour l'implantation des différents pairs. 3DEvent reprend presque les mêmes choix techniques que 3DState. L'implantation de l'intergiciel a nécessité le développement spécifique de la bibliothèque webrtc-eventstore pour pouvoir proposer une interface homogène dédiée à la synchronisation du journal d'événements en utilisant WebRTC sur deux types d'instances différentes.

Testabilité La testabilité des composantes de l'application est aisément envisageable car ils sont faiblement couplés entre eux. Tant côté côté traitement que User Interface (UI), chaque composant est isolé des autres. La séparation des responsabilités entre les différents objets du domaine portée par le DDD est un avantage conséquent pour tester les différents composants. De plus, le caractère fonctionnel de l'ES permet de faire des simulations illimitées [Erb et Kargl, 2014] sans perturber le fonctionnement de l'application. De plus

l'implantation d'un intergiciel P2P en sur-couche sur NodeJS ou le navigateur peut permettre de simuler une multitude de faux utilisateurs pour tester la robustesse et le passage à l'échelle de l'application.

4.4 Conclusion du chapitre

Ce chapitre décrit l'implantation des contributions présentées dans le chapitre précédent. Le premier prototype, 3DState, est issu des premiers travaux de cette thèse. Son implantation repose principalement sur la mise en place des technologies nécessaires pour proposer une collaboration basique. L'interface proposée permet de manipuler simplement des objets 3D dans la scène sans intégrer par d'autres moyen les collaborateurs à l'exception du changement de point de vue. La gestion de la session se concentre principalement sur la mise en relation des utilisateurs et la récupération du contenu. Le client est donc très sollicité à chaque modification. L'utilisateur est assuré que ses messages envoyés sont reçus et grâce au mécanisme de verrouillage des objets sélectionnés, les conflits sont nuls. Le transfert par différentiel d'état relève d'un calcul assez faible pour le client.

Le second prototype, 3DEvent, intègre un intergiciel P2P permettant de faire communiquer différent types de plateformes entre elles grâce à l'implantation de webrtc-eventstore. Les différences entre les Instances Serveur, qui utilisent node-webrtc sur NodeJS, et les Instances Web, qui utilisent PeerJS sur le navigateur, sont alors abstraites par cette bibliothèque qui fait l'interface et leur permet de communiquer en P2P. Cette solution technique ouvre de nouvelles perspectives quant à l'extension du P2P dans les navigateurs, les mobiles, et les systèmes embarqués. Les propositions liées à l'interface orientée tâches sont en accord avec le modèle événementiel et permettent de focaliser les interactions des utilisateurs directement sur les tâches métier, c'est-à-dire les tâches de manipulation et de visualisation 3D collaborative. La synchronisation et la gestion de la cohérence dans un environnement décentralisé comme 3DEvent sont mis en place accompagnés de modules de sensibilisation à la collaboration comme la sélection fantôme.

Les deux prototypes ont été réalisés dans le but (i) de montrer la faisabilité des propositions liées aux contributions et (ii) de proposer des expérimentations utilisateurs pour évaluer et valider les choix de conception et d'implantation des modèles utilisés.

Chapitre 5

Études utilisateurs sur l'assemblage collaboratif d'objets 3D

Sommaire

5.1 Cas d'étude : Assemblage collaboratif d'objets 3D dans un environnement web 108
5.2 Expérimentation 1 : preuve de faisabilité 109
5.2.1 Présentation de l'expérimentation 1 109
5.2.2 Résultats et discussion 111
5.2.3 Conclusion de l'expérimentation 1 112
5.3 Expérimentation 2 : intégration du framework événementiel . 113
5.3.1 Présentation de l'expérimentation 2 113
5.3.2 Résultats et discussion 115
5.3.3 Conclusion de l'expérimentation 2 120
5.4 Comparaison entre l'expérimentation 1 et l'expérimentation 2 121

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'applications (Conception Assistée par Ordinateur (CAO) collaborative, Business Information Modeling (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. Cette section présente le cas d'étude des expérimentations de cette thèse. 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]. Le prototype 3DState, sur lequel est effectué cette expérimentation, est décrit dans la Section 4.2. L'expérimentation s'intéresse à 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]. Le prototype 3DEvent, sur lequel est effectué cette expérimentation, est décrit dans la Section 4.3. L'expérimentation s'intéresse à démontrer l'intérêt d'utiliser une EDA dans le cadre de la collaboration 3D pour la visualisation et la manipulation d'objets 3D : pour les utilisateurs qui ont accès à une interface dédiée aux tâches à effectuer, pour les expérimentateurs qui peuvent analyser les interactions métier facilement et pour les performances de l'application qui repose sur un système faiblement couplé, où chaque composant concentre ses efforts sur une tâche précise et isolée.

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 ;

— 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 : la collaboration d'un

point de vue réseau, était-elle transparente pour l'utilisateur ? adaptée ? plus efficace qu'avec l'architecture de l'expérimentation 1 ?

5.2 Expérimentation 1 : preuve de faisabilité

Cette expérimentation s'inspire du cas d'étude présenté dans la section précédente pour proposer l'évaluation du prototype 3DState (Section 4.2). L'expérimentation est une étude utilisateur qui a pour objectif d'évaluer l'utilisabilité de l'architecture de communication « orientée états » dans une application d'assemblage d'objets 3D collaborative : 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) et propose de répondre aux questions suivante : quel est l'apport d'une architecture hybride dans la collaboration temps réel ? Comment l'utilisateur s'approprie-t-il le prototype mis à disposition ? Est-ce que la collaboration rend la réalisation de la tâche plus rapide et / ou plus efficace ? Un des objectifs de cette expérimentation est de montrer que le prototype réalisé sur la base d'une architecture hybride facilite la collaboration (échanges de mises à jour) sans perturber les habitudes des utilisateurs.

L'expérimentation est basée sur le prototype d'application 3DState développé selon l'implantation présentée dans la Section 4.2. 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 la Section 3.3.2.

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 téléversées par les utilisateurs. L'image correspond à l'assemblage final à obtenir et indique les différentes parties qui le composent (type et quantité). Les épreuves s'arrêtent lorsque les participants le notifient à l'expérimentateur, 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 appareil.

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 3D (sélection, rotation, translation, homothétie) et les disposer de manière à atteindre l'objectif.

Population L'expérience a été conduite sur trois groupes composés respectivement de deux, quatre et quatre participants chacun (cf. Tableau A.4) ; le dernier groupe a effectué deux épreuves. Une épreuve sur le modèle *Wind Turbine* a été effectuée avec six participants en simultané pour observer le comportement du système dans un cas non prévu. Durant l'expérimentation, les utilisateurs étaient sur le même réseau que le serveur (Local Area Network (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 (10 minutes)
   (a) explication du contexte et de l'expérimentation
   (b) prise en main du système, familiarisation avec l'interface
2. Phase de collaboration (10 minutes / 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 (10 minutes) : 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 dans *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. Les modalités d'interaction entre les participants sont observées 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 différents 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 s'explique principalement par le fait qu'ils ont atteint les objectifs fixés 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 ont même souhaité 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 la forme 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 de 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 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 à la volée 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 connexions 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 différentiels 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 clients 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 permettant d'importer des géométries, leur transmission dans le réseau P2P a causé des latences sur les pairs destinataires (gel de la fenêtre). Afin de réduire la latence lors de la transmission de larges scènes, 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'une topologie maillée partiellement et d'une distribution des modèles incluant les pairs permettraient de donner plus de responsabilité aux pairs dans la transmission des données en désengorgeant un peu le serveur et surtout le réseau P2P.

Concernant l'amélioration des fonctionnalités proposées par l'IU, 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 à nouveau utilisé dans le cadre de l'expérimentation 2 : plusieurs participants sont réunis dans l'environnement collaboratif du prototype 3DEvent (Section 4.3) pour assembler différentes parties d'un modèle. Ces travaux sont réalisés sur la base des contributions du modèle événementiel et de l'architecture de communication orientée événements 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) ?

5.3.1 Présentation de l'expérimentation 2

Cette expérimentation a été réalisée sur la base du prototype 3DEvent présenté dans la Section 4.3. Dans ces contributions, la fiabilité du modèle et de son implantation est soulignée de deux manières :

— fonctionnelle : manipulation et visualisation 3D, historique ;

— et technique : récupération de l'information, cohérence des données, disponibilité du réseau.

L'observation du comportement des participants est effectuée de manière quantitative (outils de surveillance) et de manière qualitative (questionnaire en Annexe A.4.2) lors de l'exécution d'une tâche coopérative au sein de l'application.

Tâche à effectuer Les participants devaient assembler les différentes parties d'un modèle 3D en utilisant le prototype 3DEvent (Section 4.3) 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 faciles de l'application, il a été choisi de conserver des manipulations 3D courantes de haut niveau pour que les utilisateurs les prennent en main rapidement.

Population L'expérience a été conduite sur six groupes de deux ou trois participants chacun (trois groupes de deux et trois groupes de trois). Chaque participant était localisé en France dans une zone urbaine disposant de bonnes infrastructures avec une bonne connexion internet (au moins 20Mb/s). L'utilisation d'un réseau Wide Area Network (WAN) de bonne qualité permet d'éviter d'avoir des latences extrêmes (>10 secondes) au cours des expérimentations tout en étant dans un contexte réaliste. 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 3D utilisés lors de l'expérience sont décrits dans le Tableau A.5. L'expérimentation se déroule en trois phases :

1. Phase *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 *Collaboration* : Un groupe de participants réalise deux assemblages sur
   — un petit modèle (10 ou 12 parties),
   — puis un plus gros modèle (16 parties).

   La phase de collaboration a été effectuée 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 sur chaque pair (incluant chaque participants). 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 réalisation, 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).

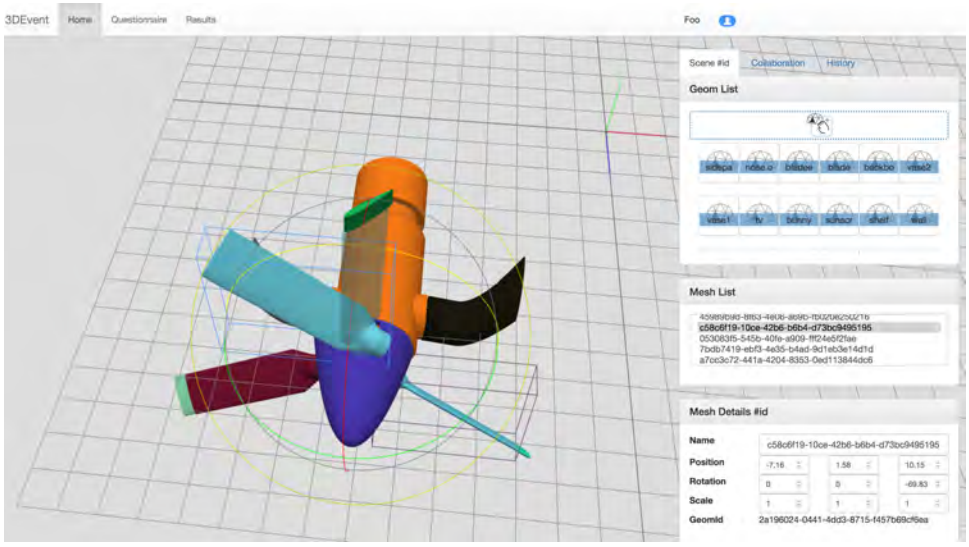
Le questionnaire est inspiré de [Lewis, 1995], qui permet d'évaluer la facilité d'utilisation du système et l'implication dans la collaboration de chacun des participants. Une échelle de notation sur sept points a été utilisée, (1 : pas d'accord ; 7 : d'accord) pour avoir assez de points de discrimination [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

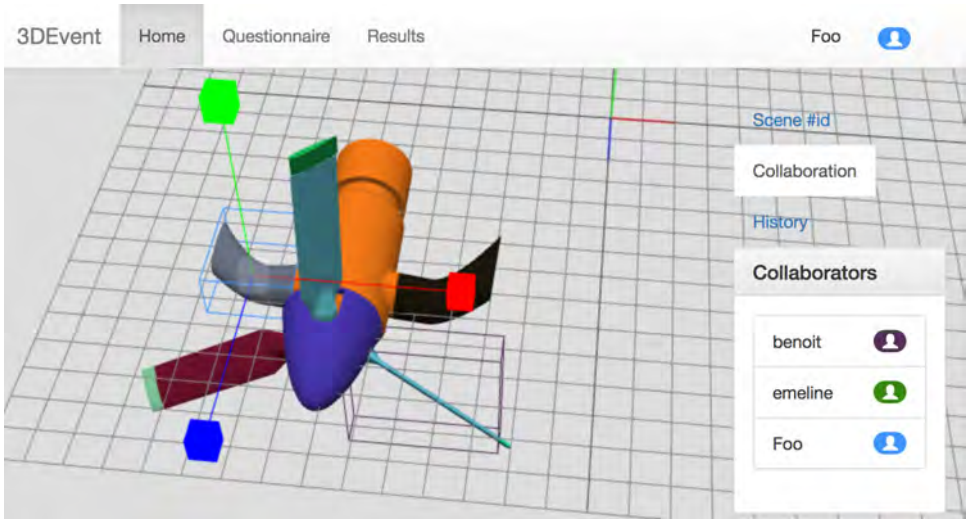
La Figure 5.1 montre quelques captures d'écran durant une session collaborative sur le modèle *Rotor* : la Figure 5.1a montre l'environnement de l'utilisateur alors qu'il est en train de faire une rotation sur une des pales et la Figure 5.1b montre la réalisation d'une mise à l'échelle dans l'environnement 3D.

La Figure 5.2 est une capture d'écran de l'interface Event Store qui montre les premiers événements enregistrés dans la base de données. Tous les événements enregistrés sont nommés selon la version et le mot clé de l'agrégat concerné (ex : 18@scene-turbine). Cela permet de catégoriser les événements et de faciliter leur requête.

Analyse des interactions Les traces des utilisateurs récupérées au cours des expérimentations sont la base du travail d'analyse présenté ci-après. Ces traces, composées d'événements générés par les utilisateurs, permettent de savoir qui (Figure 5.3a) a fait quoi (Figure



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

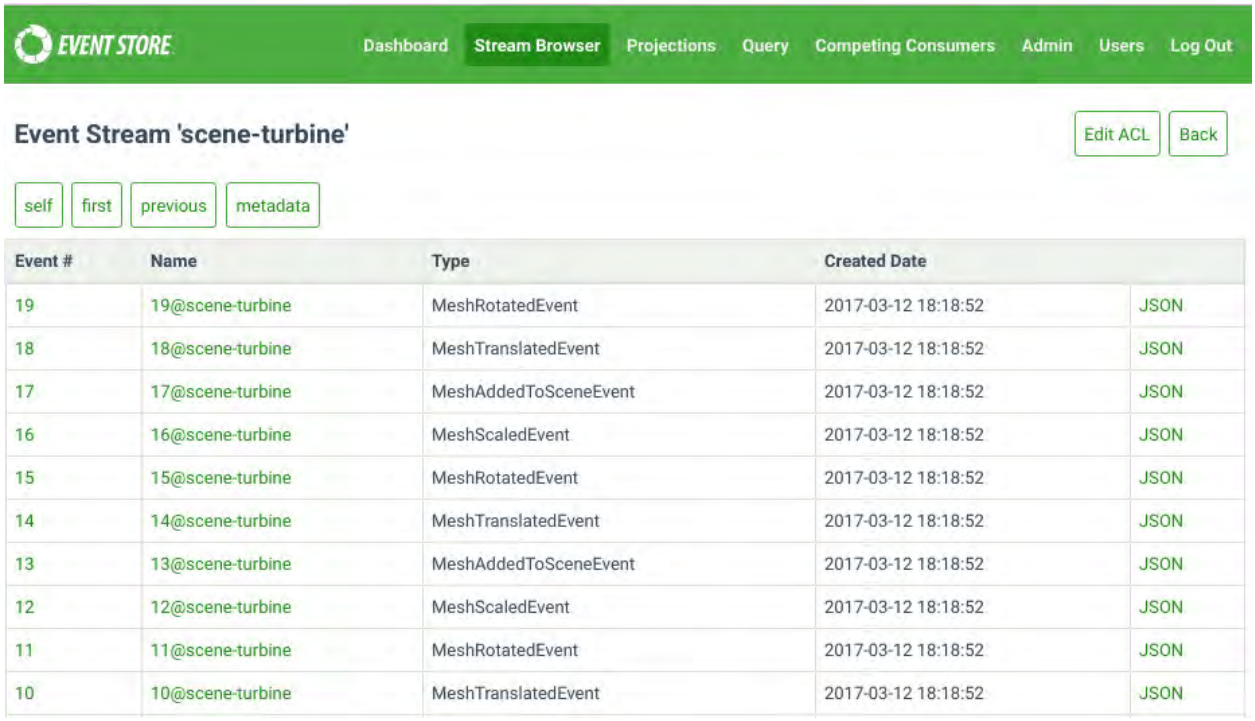


(b) Mise à l'échelle (environnement 3D) et liste des collaborateurs (panneau latéral)

FIGURE 5.1 – Interface utilisateur pendant une session collaborative (trois personnes) Les boîtes englobantes représentent la sélection des différents collaborateurs pendant la session.

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.

La 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*).

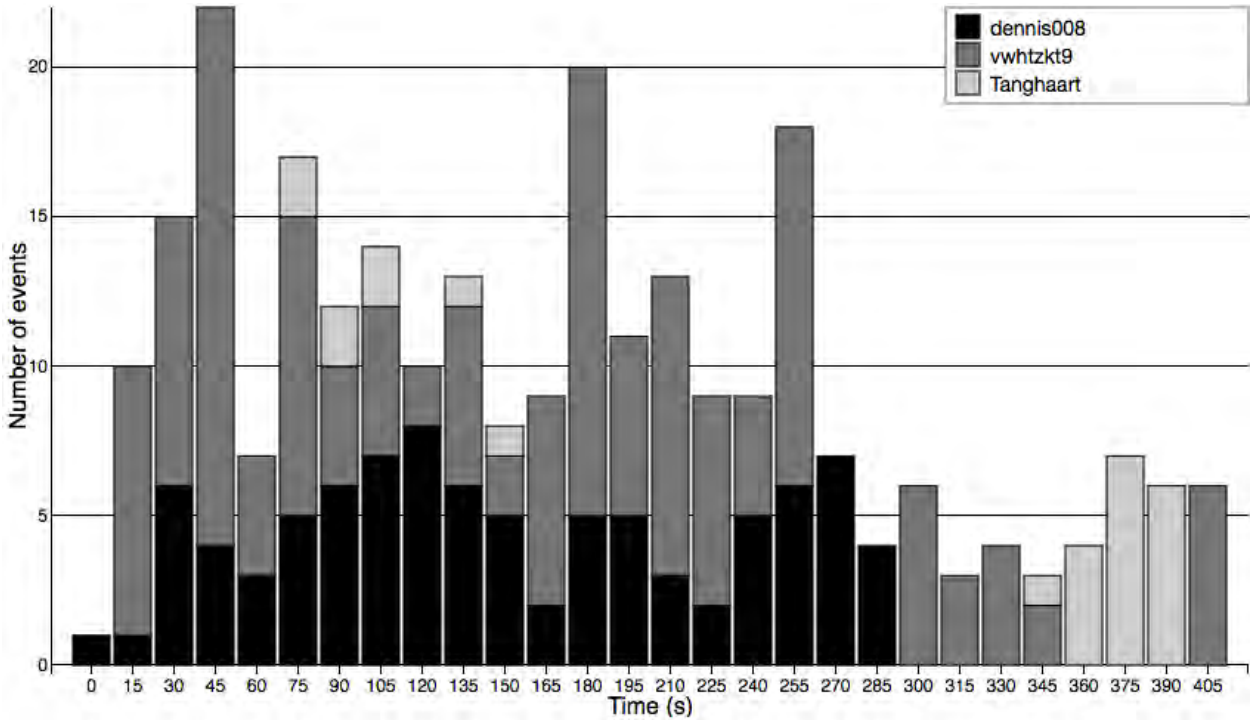


Event #	Name	Type	Created Date	
19	19@scene-turbine	MeshRotatedEvent	2017-03-12 18:18:52	JSON
18	18@scene-turbine	MeshTranslatedEvent	2017-03-12 18:18:52	JSON
17	17@scene-turbine	MeshAddedToSceneEvent	2017-03-12 18:18:52	JSON
16	16@scene-turbine	MeshScaledEvent	2017-03-12 18:18:52	JSON
15	15@scene-turbine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

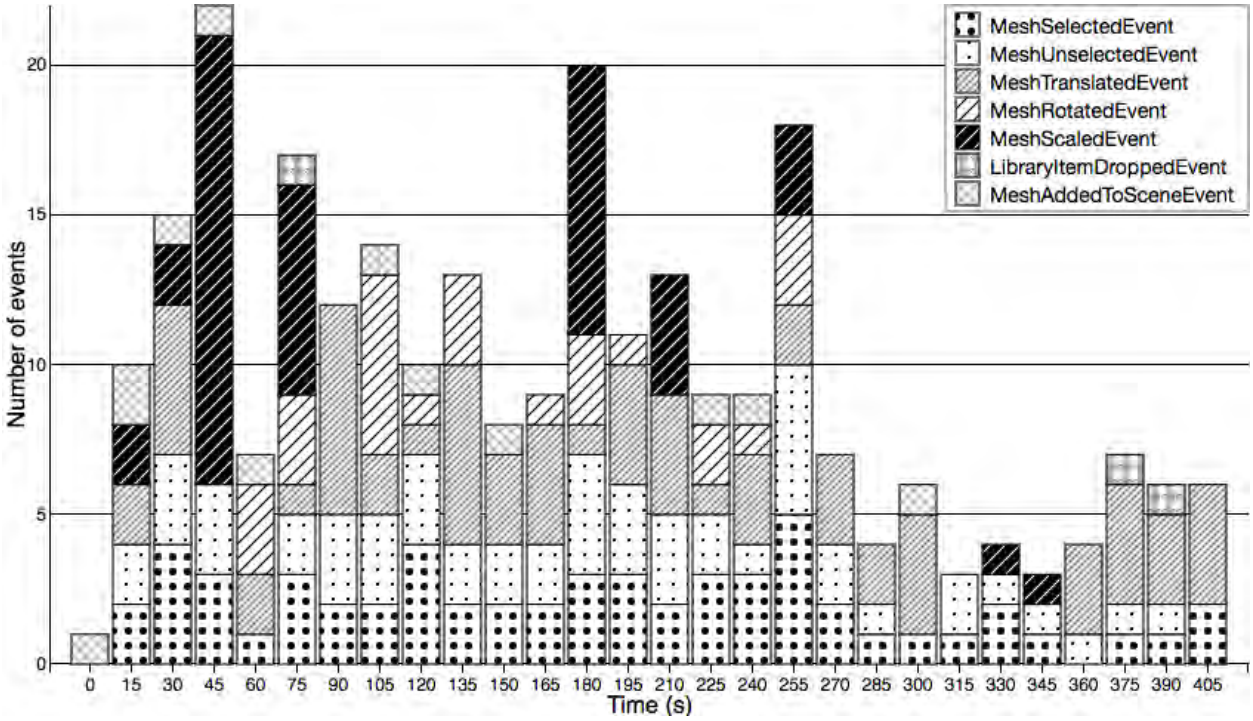
FIGURE 5.2 – Persistance long-terme (Event Store®), base de données/outil de monitoring

Seul un utilisateur a ajouté un objet en utilisant la fonctionnalité pour déposer un objet à un endroit spécifique de la scène (*meshDropped*). Les nombres d'événements concernant la sélection et la désélection sont à peu près similaire. La différence s'explique par le fait que l'événement de désélection n'est pas déclenché lorsque l'utilisateur change d'objet sélectionné. En effet, 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 (causé par le 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 revienne 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 qui correspondent aux 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 der-



(a) Par utilisateur



(b) Par type d'événement

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

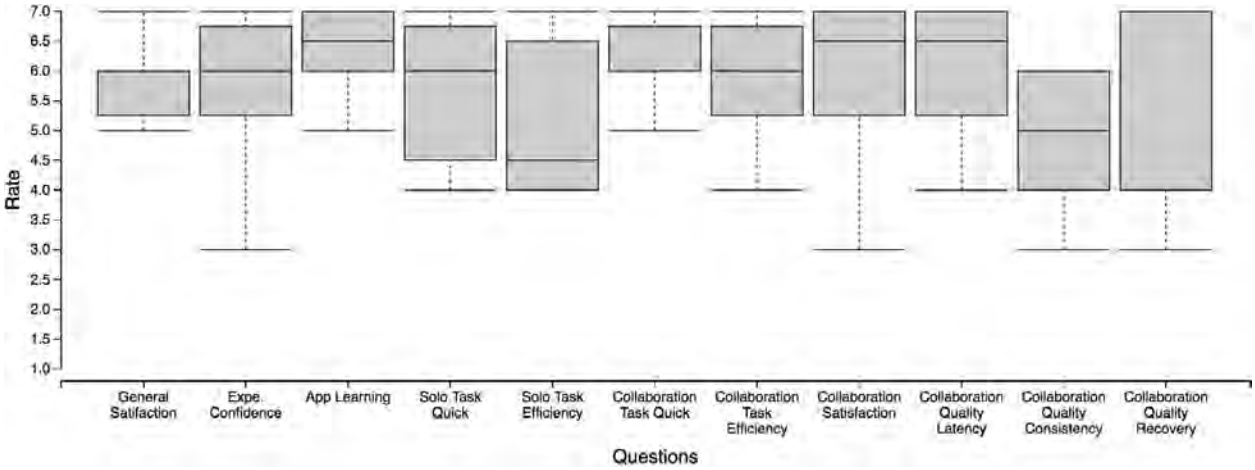


FIGURE 5.4 – Résultats des questionnaires collectés

nier aspect est illustré par un exemple dans la Figure 5.3b à 45s, où un nombre élevé de MeshScaledEvent est détecté. *A posteriori*, il est possible de constater 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 de la phase *Solo* et des phases *Collaboration* 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 se figurer le profil essentiel des résultats des mesures quantitatives effectuées.

Globalement, les tâches ont été réalisées plus rapidement et plus efficacement de manière collaborative (*Collaboration*) que seul (*Solo*). La facilité d'utilisation et la simplicité de l'interface sont également soulignées positivement par plusieurs utilisateurs.

Parmi les retours d'expérience négatifs, l'instabilité 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 sans effets de bord, leur permettant de coopérer efficacement, avec peu de conflits détectés.

Durant l'une des expérimentations en phase *Collaboration*, 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. Sur le réseau, la détection des conflits (mauvaise version, désynchronisation), la politique en place consiste à reconstruire l'état de

l'application avant le conflit, i.e. jusqu'au dernier événement qui n'est pas en conflit, pour pouvoir resynchroniser les utilisateurs sur une base stable. Cette mécanique qui peut sembler envahissante n'a pas dérangé les utilisateurs lors de la modélisation. Pour pallier l'absence de résolution automatique de conflit, les utilisateurs ont été mis a contribution dans les cas de désaccords (opérations opposées sur le même objet) : la résolution du conflit passe par un canal externe (chat) pour que les utilisateurs se mettent d'accord. Cette connaissance du métier n'est donc pas intégrée au système.

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 notamment lors des désynchronisation, aucun affichage ne prévient l'utilisateur ou encore la sensibilisation à l'historique des objets manipulés. Pour modérer ces résultats, il est important de rappeler que les modèles utilisés ne sont pas très complexes et que tous les participants étaient débutants sur le système et parfois néophytes en modélisation 3D. 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 plateformes. 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 peuvent être des indicateurs sur le fait que les participants ont apprécié positivement les épreuves proposées durant l'expérimentation. 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

Les différentes phases de l'expérimentation amènent le participant à prendre en main rapidement le prototype (Section 4.3.2). Le protocole, qui en proposant plusieurs épreuves sur différents modèles, offre 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 choisi lors de l'implantation du modèle événementiel. Les utilisateurs ont globalement exprimé leur satisfaction

concernant leur expérience générale : objectifs atteints, interface facile à prendre en main, collaboration facilitant la réalisation de la tâche. Concernant l'architecture 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-utilisateur du même agrégat et sont donc plutôt rares. De plus, les utilisateurs étant sensibilisés (par exemple : les boîtes englobantes de sélection en couleur) au fait qu'un autre utilisateur a déjà sélectionné l'objet, ils sont ainsi prévenus de l'occurrence potentielle de 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, 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.

Chapitre 6

Conclusion

Sommaire

6.1 Discussion 124
6.2 Potentielles applications et autres utilisations 128
6.3 Perspectives 129

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.

6.1 Discussion

Pour répondre aux différentes questions de recherche posées en introduction, plusieurs contributions ont été proposées, implantées et soumises à des expérimentations pour adresser la problématique de cette thèse : comment engager toutes les ressources à disposition lors de l'édition collaborative d'environnements 3D sur le web ?

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, en intégrant un modèle de cohérence des données.

L'architecture de communication 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'états. Bien que la politique de connectivité ne permette 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, configuré comme fiable et ordonné, et par le mécanisme de verrouillage évitant l'édition concurrente d'objets 3D.

Partant de ces observations, la seconde contribution liée à l'architecture de communication repose sur le paradigme événementiel. Une politique de connectivité plus souple permet aux pairs de devenir des 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 sur 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 conflits 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 pour une dissémination plus stable et rapide. Selon le nombre d'utilisateurs, 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. L'historique et le « défaire / refaire » sont des fonctionnalités 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 s'adapte par ses capacités à permettre une forte traçabilité des changements qui se produisent au sein du système.

La modélisation logicielle 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), le système impose plusieurs niveaux de vérification des données : de la saisie utilisateur à la validation les règles métiers. 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 tâches plus proche des attentes métiers de l'utilisateur. La projection découple les flux de la base de données des événements servant une ou plusieurs vues. En effet, son rôle d'intermédiaire est de 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 mené 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 métiers 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 la dépendance du client au serveur pour valider le contenu créé. L'utilisateur a accès localement aux événements. Il peut ainsi générer de nouveaux événements sur la base de ceux qu'il

possède déjà. Les projections fonctionnent également sur ces événements stockés. Le cycle des données est donc complètement géré en interne et libère l'utilisateur d'une forte liaison avec la base de données centrale. La synchronisation des données produites hors ligne est effectuée à la reconnexion du pair au réseau. En effectuant la différence entre les agrégats locaux et distants, les répliques sont capables de synchroniser leur Event Store.

L'autre partie de la réponse à la **QR 3** est apportée par la contribution portant sur la présentation d'une architecture de communication hybride. Le système renforce l'autonomie des utilisateurs en leur permettant de s'échanger directement des informations. Le serveur n'est plus une source d'engorgement pour l'accès à la base de données, mais un ensemble de pairs reliés à une source de vérité de l'application. Les pairs liés à la base de données sont répartis dans le réseau de manière à être plus disponibles et proches pour les pairs sur lesquels les utilisateurs travaillent. En cas d'arrêt temporaire de quelques-uns de ces pairs, l'application peut continuer de fonctionner en collaboration sur les pairs restants. La mise en place d'un environnement collaboratif nécessite également un mécanisme de gestion de conflits. Cette thèse présente successivement deux implantations de ce mécanisme, l'un pessimiste (verrou), l'autre optimiste (*Last Write Wins*). Considérant que le premier peut générer des effets de bord considérables dans l'environnement et amener à la frustration des utilisateurs, le second, plus permissif et accompagné de mécanismes de sensibilisation au groupe (boîtes englobantes) et de sensibilisation à l'interaction (sélection fantôme), renie ces inconvénients et laisse plus de liberté à l'utilisateur dans ses choix de sélection, et plus largement, d'interaction.

La **QR 4** s'intéresse à l'implémentation d'un EVC 3D sur le web. La question s'oriente principalement sur les choix qu'impose ce type d'environnement très contraint, tant au niveau du métier de la modélisation 3D collaborative qu'au niveau des technologies et des techniques qui s'y rapportent. Les prototypes web sont réalisés à partir de technologies issues de standards (définis par World Wide Web Consortium (W3C), Internet Engineering Task Force (IETF) et ECMA pour JS) largement supportés comme WebGL pour le rendu de l'environnement 3D et parfois encore en évolution comme WebRTC DataChannel pour la transmission de données en P2P. Dans le prototype reposant sur le paradigme événementiel, l'implantation d'une IU orientée tâches rend possible un accompagnement de l'utilisateur dans chaque action, en lui procurant une interface adaptée à la tâche en cours de réalisation. Le pendant de ce découpage de l'interface par tâches est un passage à l'échelle, facilité par la production de vues matérialisées issues de projections d'événement qui peuvent être mises en cache. Une projection pouvant servir plusieurs vues, il est possible de décliner plusieurs visualisations des mêmes données. La projection liée à l'environnement 3D peut, par exemple, proposer plusieurs déclinaisons des mêmes données pour chaque vue qui en dépend.

Cela peut se traduire par un rendu effectué par des bibliothèques 3D différentes (Three.JS, BabylonJS) ou un rendu adapté selon un profil utilisateur. Les projections sont également un facteur facilitant l'évolution de l'application, simplifiant le processus d'agrégation et de filtrage des événements.

La réalisation de plusieurs prototypes fonctionnant sur différents paradigmes pour les expérimentations a permis de mettre en évidence les avantages et inconvénients de ceux-ci. Passer d'un paradigme orienté état à un paradigme événementiel a apporté plusieurs bénéfices liés au métier, notamment en ce qui concerne la conservation de l'intention de l'utilisateur au moment de la réalisation d'une action. En effet, les événements sont auto-descriptifs et sont également porteurs du lien de causalité qui reflète cette intention. Le réseau, responsable de la dissémination de ces informations, ne porte pas la responsabilité de la conservation de cette propriété, c'est à la couche applicative (aux projections) de s'en charger. Cela permet d'alléger les échanges avec des connexions P2P ayant une configuration non fiable et non ordonnée. Le protocole d'échanges proposé repose sur une politique push / pull qui s'assure de la synchronisation des différents pairs. En cas de conflits, les participants sont notifiés par le gestionnaire de conflits, ce qui leur permet de relancer la synchronisation sur une base commune en cas de problème insolvable. Le ré-ordonnancement passe par le mécanisme de visualisation flexible. Les événements sont publiés a destination des projections une fois reçu et validés. Seuls les événements en séquence (pas d'événements manquants, de « trou ») sont transmis aux projections, ce qui permet de stocker en avance les futurs événements et de déclencher une nouvelle demande des événements manquants. Cela permet de proposer rapidement un retour visuel à l'utilisateur d'une scène en cours de synchronisation par exemple même si tous les événements ne sont pas réceptionnés pour qu'il commence à percevoir, s'immerger voire travailler sur la scène.

Enfin, l'expérimentation 2 a directement bénéficié de l'implantation d'un modèle orienté événements pour le prototype utilisé : l'intégration du métier a permis l'observation minutieuse du travail réalisé par les utilisateurs au sein de l'environnement. C'est la base du travail d'analyse des interactions de cette expérimentation.

Dans la **QR 5**, la volonté de trouver des métriques adaptées à l'évaluation d'un EVC 3D est exprimée tant pour ce qui concerne le réseau que la collaboration. La question se pose à la fois en termes quantitatifs et qualitatifs. Les expérimentations détaillées dans cette thèse présentent plusieurs facteurs d'évaluation qualitatifs. En effet, les deux expérimentations portent sur une étude des utilisateurs dans le cas d'application de l'assemblage de modèles 3D de manière collaborative pour la visualisation et la manipulation d'objets dans un environnement web. La mise en place des protocoles d'expérimentation a permis de soumettre les utilisateurs à différentes épreuves d'assemblage : contraint par un objectif déterminé,

création libre. L'évaluation a porté sur les observations de l'expérimentateur et des réponses données par les participants au questionnaire concernant différents aspects : le ressenti du participant par rapport à la latence dans l'environnement, la gestion de la cohérence, la qualité de la récupération des données en cas de déconnexion, mais également sur l'apprentissage et la facilité d'utilisation de l'application. La qualité de la collaboration est également soumise au questionnaire pour savoir si la coopération améliore l'efficacité et / ou la vitesse de réalisation de la tâche. Les contraintes liées à ce type d'expérimentations sont nombreuses : nombre de participants, taille des données 3D, gestion de la cohérence... Les métriques quantitatives sont seulement abordées dans cette thèse sous l'angle de l'implantation d'outils de surveillance.

6.2 Potentielles applications et autres utilisations

La conception d'un modèle événementiel à travers l'implémentation d'une plateforme comme 3DEvent peut servir d'autres applications asynchrones, distribuées et orientées événements.

— *Git-like app.* Les solutions pour faire de la gestion de version, comme par exemple MeshGit [Denning et Pellacini, 2013] 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 notamment à cause du coût et de la complexité que cela peut induire dans des architectures traditionnelles ; 3DEvent permet justement de 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 crowdsouring.* 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.

6.3 Perspectives

Les évaluations présentées sont à compléter avec des évaluations quantitatives permettant de comparer les architectures de communication hybrides exposées avec d'autres systèmes pour la modélisation 3D collaborative.

Virtualisation des comportements collaboratifs La simulation de comportements collaboratifs sous forme de scénarios enregistrés (grâce à l'ES) aurait pu permettre d'obtenir des résultats sur des métriques quantitatives fiables telles que la quantité de données transmises, la variation de la topologie du réseau. Alors que des auteurs sont capables de réunir jusqu'à cent personnes simultanément sur une expérimentation pour montrer le passage à l'échelle de leur système [Hu et Chen, 2017], il semble préférable pour des questions de temps, de logistiques et de coûts, d'évoquer les possibilités de virtualisation offertes par les plateformes réceptives au standard WebRTC pour tester les capacités du système [Haque *et al.*, 2016].

Pour cela il est nécessaire d'avoir un système de virtualisation du réseau pour pouvoir faire varier les différents critères – nombre d'utilisateurs, nombre de messages, nombre d'événements traités, taille des données 3D (impact sur la visualisation), longueur des scénarios – qui indiqueraient de manière quantitative la capacité du système à passer à l'échelle lors de la collaboration. Le temps de synchronisation semble être un bon indicateur concernant cette propriété. Ces données pourraient permettre d'appuyer les résultats qualitatifs qui sont présentés dans cette thèse. Les outils liés à WebRTC sont encore au stade expérimental mais produisent déjà des statistiques par rapport à quelques un des critères énoncés sur les données transitant sur une connexion DataChannel.

Compatibilité et plasticité Le standard WebRTC, sur lequel repose la couche P2P, est encore en évolution, parfois instable, hétérogène [1] ou indisponible, c'est pourquoi en proposant l'utilisation de protocoles alternatifs à WebRTC (WebSocket, HTTP, ORTC) l'accessibilité est renforcée. En fonction du type de connexion et d'appareil, l'application pourrait basculer d'un protocole à l'autre pour stabiliser la connexion en choisissant selon différents facteurs (énergétique, bande passante) indiqués par l'appareil ou l'utilisateur. Le type de modélisation peut également s'adapter en fonction de ces critères pour ne pas dégrader trop l'expérience utilisateur selon les cas d'utilisation en proposant par exemple un rendu image du modèle 3D effectué sur un serveur [Abdallah *et al.*, 2014] ou réalisé par une grappe de pairs [Li *et al.*, 2015]. Ces perspectives d'adaptation du rendu 3D, ou plus généralement de l'interface font référence à un intérêt pour la plasticité dans les interfaces 3D [Lacoche *et al.*, 2014] en s'appuyant sur les métadonnées issues de l'utilisateur, du système, de l'appareil, de la connexion et de la sémantique du contenu produit et des contraintes de temps liées au cas d'utilisation.

Simplification et compression de maillage Si la plateforme web est accessible depuis n'importe quel appareil possédant un navigateur, la transmission (le téléversement ou le téléchargement) de modèles 3D peut s'avérer long et fastidieux. Afin de réduire les latences sur des scènes plus large, la transmission des données 3D pourrait être améliorée grâce à un format de compression propre aux événements produits. Grâce au réseau P2P il est possible de rapprocher les ressources des utilisateurs selon leurs intérêts. Le rendu assisté par les pairs [Zhu *et al.*, 2011] couplé avec une couche dédiée au routage par sujet [Baldoni *et al.*, 2007] pourrait encourager un meilleur équilibre dans la répartition de la charge de transmission.

Un nouveau modèle en cours d'élaboration reprend les principes du modèle événementiel proposé dans cette thèse pour introduire dans l'architecture de communication, le principe de « nœud responsable ». En rendant un nœud responsable d'un agrégat, il devient alors l'unique garant de sa cohérence, contrairement au système de consensus actuellement en place. Toutes les modifications de l'agrégat doivent alors impérativement passer par ce nœud pour être validées (routage basé sur le contenu). En cas de départ du réseau, un nouveau nœud prend la responsabilité de l'agrégat (par exemple : lors de la sélection de l'objet dans la scène). Ce principe de « nœud responsable » permettrait, dans une situation où les modifications localisées sur un même objet sont courantes, d'isoler la responsabilité de la cohérence pour chaque agrégat.

1. Les standards ne sont pas toujours respectés...

Bibliographie

[Abdallah *et al.*, 2014] Abdallah, A., Fryazinov, O., Adzhiev, V. et Pasko, A. (2014). 3D Web-Based Shape Modelling : Building up an Adaptive Architecture. (c):96–102. 6.3

[Abdullin, 2011] Abdullin, R. (2011). Event Sourcing - Projections. 3.2.4

[Avram et Marinescu, 2006] Avram, A. et Marinescu, F. (2006). *Domain-Driven Design Vite fait*. 2.6

[Baldoni *et al.*, 2007] Baldoni, R., Ruberti, S. a., Querzoni, L. et Tucci-piergiovanni, S. (2007). TERA : Topic-based Event Routing for peer-to-peer. *International Conference on Distributed Event-Based Systems (DEBS)*, pages 2–13. 2.3.2, 6.3

[Banavar *et al.*, 1999] Banavar, G., Chandra, T., Mukherjee, B., Nagarajarao, J., Strom, R. et Sturman, D. (1999). An efficient multicast protocol for content-based publish-subscribe systems. *Proceedings. 19th IEEE International Conference on Distributed Computing Systems (Cat. No.99CB37003)*, pages 262–272. 2.3.2

[Bang *et al.*, 2017] Bang, J., Brun, Y. et Medvidovic, N. (2017). Continuous Analysis of Collaborative Design. 2.3.1

[Bang *et al.*, 2010] Bang, J. Y., Popescu, D., Edwards, G., Medvidovic, N., Kulkarni, N., Rama, G. M. et Padmanabhuni, S. (2010). CoDesign : a highly extensible collaborative software modeling framework. *2010 ACM/IEEE 32nd International Conference on Software Engineering*, 2:243–246. 2.3.1

[Baran, 2015] Baran, I. (2015). Under the Hood : How Collaboration Works in Onshape. 2.1.2, 2.2

[Bauwens, 2011] Bauwens, M. (2011). Collaboration - P2P Foundation. 1.1.1

[Becher, 2012] Becher, M. (2012). Interactive Volume Visualization with WebGL. (06). 2.1.2

[Behr *et al.*, 2010] Behr, J., Jung, Y., Keil, J., Drevensek, T., Zöllner, M., Eschler, P. et Fellner, D. W. (2010). A Scalable Architecture for the HTML5/X3D Integration Model X3DOM. *Proceedings of the 15th International Conference on Web 3D Technology*, 1(212):185–194. 2.1.2

[Bentley et Wakefield, 1997] Bentley, P. et Wakefield, J. (1997). Generic Evolutionary Design. *Soft Computing in Engineering Design ...*, pages 1–10. 1.1.4

[Birman et Joseph, 1987] Birman, K. et Joseph, T. (1987). Exploiting virtual synchrony in distributed systems. *ACM SIGOPS Operating Systems Review*, 21(5):123–138. 2.3.2

[Bonér *et al.*, 2014] Bonér, J., Farley, D., Kuhn, R. et Thompson, M. (2014). The Reactive Manifesto. 2.3.5

[Brown *et al.*, 2003] Brown, D., Julier, S., Baillot, Y. et Livingston, M. (2003). An event-based data distribution mechanism for collaborative mobile augmented reality and virtual environments. *IEEE Virtual Reality, 2003. Proceedings.*, 2003. 1.1.4

[Buford *et al.*, 2009] Buford, J., Yu, H. et Lua, E. K. (2009). *P2P Networking and Applications (Google eBook)*. 3.3.1

[Cai et Yu, 2014] Cai, G. et Yu, B. (2014). Event-based awareness promotion for distributed collaborative activities. *2014 International Conference on Collaboration Technologies and Systems, CTS 2014*, pages 302–309. 2.3.1

[Calabrese *et al.*, 2016] Calabrese, C., Salvati, G., Tarini, M. et Pellacini, F. (2016). cSculpt : a system for collaborative sculpting. *ACM Transactions on Graphics*, 35(4):1–8. 2.2

[Callahan *et al.*, 2008] Callahan, S., Schenk, M. et White, N. (2008). Building a collaborative workplace. *Anecdote : putting stories to work*, pages 1–11. 1.1.1

[Carzaniga *et al.*, 2000] Carzaniga, A., Rosenblum, D. S., Science, C. et Wolf, A. L. (2000). Achieving Scalability and Expressiveness in an Internet-Scale Event Notification Service. *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, pages 219–227. 2.3.2

[Carzaniga et Wolf, 2002] Carzaniga, A. et Wolf, A. L. (2002). A Benchmark Suite for Distributed Publish / Subscribe Systems. *Program*. 2.3.2

[Castro *et al.*, 2002] Castro, M., Druschel, P., Kermarrec, A. M. et Rowstron, A. I. T. (2002). Scribe : A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, 20(8):1489–1499. 2.3.2

[Chandrasegaran *et al.*, 2013] Chandrasegaran, S. K., Ramani, K., Sriram, R. D., Horváth, I., Bernard, A., Harik, R. F. et Gao, W. (2013). The evolution, challenges, and future of knowledge representation in product design systems. *Computer-Aided Design*, 45(2):204–228. 1.1

[Chandy *et al.*, 2011] Chandy, M. K., Etzion, O. et Ammon, R. V. (2011). *The event processing manifesto*. Numéro 10201. 1.1.4, 3.2

[Chen et Hou, 2014] Chen, H.-M. et Hou, C.-C. (2014). Asynchronous online collaboration in BIM generation using hybrid client-server and P2P network. *Automation in Construction*, 45:72–85. 2.2.3

[Cheng *et al.*, 2009] Cheng, W., Liu, D. et Ooi, W. T. (2009). Peer-assisted view-dependent progressive mesh streaming. *Proceedings of the seventeen ACM international conference on Multimedia - MM '09*, page 441. 4.3.1

[Cheng *et al.*, 2013] Cheng, Y., He, F., Cai, X. et Zhang, D. (2013). A group Undo/Redo method in 3D collaborative modeling systems with performance evaluation. *Journal of Network and Computer Applications*, 36(6):1512–1522. 2.3.5

[Chu et Chan, 2012] Chu, C. H. et Chan, Y. H. (2012). Optimized product data transmission in LOD-based collaborative design using P2P CAD streaming. *Journal of Intelligent Manufacturing*, 23(5):1559–1571. 4.3.1

[Cristea *et al.*, 2011] Cristea, V., Pop, F., Dobre, C. et Costan, A. (2011). Distributed architectures for event-based systems. *Studies in Computational Intelligence*, 347:11–45. 1.1.4

[Denning et Pellacini, 2013] Denning, J. D. et Pellacini, F. (2013). MeshGit. *ACM Transactions on Graphics*, 32(4):1. 6.2

[Desprat *et al.*, 2017] Desprat, C., Caudesaygues, B., Luga, H. et Jessel, J.-P. (2017). Doctoral Symposium : Loosely Coupled Approach for Web-Based Collaborative 3D Design. *In Proceedings of ACM International Conference on Distributed Event-Based Systems*. 1.3, 3.1, 3.3.4, 5.1

[Desprat *et al.*, 2015a] Desprat, C., Jessel, J.-P. et Luga, H. (2015a). A 3D collaborative editor using WebGL and WebRTC. *Proceedings of the 20th International Conference on 3D Web Technology - Web3D '15*, pages 157–158. 1.3, 3.3.2, 5.1

[Desprat *et al.*, 2016] Desprat, C., Jessel, J.-P. et Luga, H. (2016). 3DEvent : A Framework Using Event-Sourcing Approach For 3DWeb-Based Collaborative Design in P2P. *In Proceedings of the 21st International Conference on Web3D Technology - Web3D '16*, pages 73–76. 1.3, 3.1, 3.3.4, 5.1

[Desprat *et al.*, 2015b] Desprat, C., Luga, H. et Jessel, J.-P. (2015b). Hybrid client-server and P2P network for web-based collaborative 3D design. *WSCG 2015 Conference on Computer Graphics, Visualization and Computer Vision*, pages 229–238. 1.3, 2.2.2, 3.1, 3.3.2, 5.1

[Dias, 2015] Dias, D. (2015). browserCloud.js - A federated community cloud served by a P2P overlay network on top of the web platform. (May). 2.2.2

[Ekadiyanto et Hunger, 2012] Ekadiyanto, F. A. et Hunger, A. (2012). Prototype development towards hybrid peer-to-peer framework of distributed environment for cooperative and collaborative work. *2012 International Conference on Computer and Communication Engineering (ICCCE)*, (Iccce):344–348. 2.2.3

[Ellis et Gibbs, 1989] Ellis, C. A. et Gibbs, S. J. (1989). Concurrency control in groupware systems. *ACM SIGMOD Record*, 18(2):399–407. 1.1.3, 2.1.1

[Erb, 2015] Erb, B. (2015). Towards Distributed Processing on Event- sourced Graphs Towards Distributed Processing on Event-sourced Graphs Extended Abstract. (July):5–9. 2.3.5

[Erb et Kargl, 2014] Erb, B. et Kargl, F. (2014). Combining discrete event simulations and event sourcing. *Proceedings of the 7th International ICST Conference ...*. 4.3.4

[Erb *et al.*, 2017] Erb, B., Meissner, D. et Kargl, F. (2017). Chronograph - A Distributed Processing Platform for Online and Batch Computations on Event-sourced Graphs. *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems - DEBS '17*, pages 78–87. 2.3.5

[Evans, 2003] Evans, E. (2003). *Domain-Driven Design : Tackling Complexity in the Heart of Software*. Addison Wesley. 2.3.3, 2.3.3

[Fowler, 2003] Fowler, M. (2003). *Patterns of Enterprise Application Architecture*, volume 23. Addison-Wesley Longman Publishing Co., Inc. 2.3.5

[Gadea *et al.*, 2016] Gadea, C., Hong, D., Ionescu, D. et Ionescu, B. (2016). An architecture for web-based collaborative 3D virtual spaces using DOM synchronization. *2016 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, pages 1–6. 2.1.2, 3.3.2

[Gotta, 2007] Gotta, M. (2007). Categorizing Collaboration. 1.1.1

[Grasberger *et al.*, 2013] Grasberger, H., Shirazian, P., Wyvill, B. et Greenberg, S. (2013). A data-efficient collaborative modelling method using websockets and the BlobTree for over-the air networks. *Proceedings of the 18th International Conference on 3D Web Technology - Web3D '13*, page 29. 2.1.2, 2.2

[Greenberg et Marwood, 1994] Greenberg, S. et Marwood, D. (1994). Real time groupware as a distributed system. *In Proceedings of the 1994 ACM conference on Computer supported cooperative work - CSCW '94*, pages 207–217, New York, New York, USA. ACM Press. 2.1.1

[Grimstead *et al.*, 2005] Grimstead, I. J., Walker, D. W. et Avis, N. J. (2005). Collaborative visualization : A review and taxonomy. *Proceedings - IEEE International Symposium on Distributed Simulation and Real-Time Applications, DS-RT*, pages 61–69. 2.1.2

[Ha *et al.*, 2015] Ha, Y.-U., Jin, J.-H. et Lee, M.-J. (2015). Lets3D : A Collaborative 3D Editing Tool Based On Cloud Storage. *International Journal of Multimedia and Ubiquitous Engineering*, 10(9):189–198. 2.1.1

[Hand, 1997] Hand, C. (1997). A Survey of 3D Interaction Techniques. *Computer Graphics Forum*, 16(5):269–281. 1.1.2

[Haque *et al.*, 2016] Haque, S. A., Islam, S., Islam, M. J. et Grégoire, J. C. (2016). An architecture for client virtualization : A case study. *Computer Networks*, 100:75–89. 6.3

[Helmer *et al.*, 2011] Helmer, S., Poulovassilis, A. et Xhafa, F. (2011). *Reasoning in event-based distiributed systems*. 2.3.1

[Hinze *et al.*, 2009] Hinze, A., Sachs, K. et Buchmann, A. (2009). Event-based Applications and Enabling Technologies. *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, pages 1 :1–1 :15. 1.1.4, 3.2

[Hogg, 2012] Hogg, S. (2012). What About Stream Control Transmission Protocol (SCTP) ? 2.2.1

[Houston *et al.*, 2013] Houston, B., Chen, R., McKenna, T., Larsen, W., Larsen, B., Caron, J., Nikfetrat, N., Leung, C., Silver, J., Kamal-Al-Deen, H. et Callaghan, P. (2013). Clara.io. *ACM SIGGRAPH 2013 Studio Talks on - SIGGRAPH '13*, pages 1–1. 2.1.2, 2.2

[Hu *et al.*, 2008] Hu, S. Y., Huang, T. H., Chang, S. C., Sung, W. L., Jiang, J. R. et Chen, B. Y. (2008). FLoD : A framework for peer-to-peer 3D streaming. *Proceedings - IEEE INFOCOM*, pages 2047–2055. 4.3.1

[Hu et Chen, 2017] Hu, Y. et Chen, Z. (2017). WebTorrent Based Fine-grained P2P Transmission of Large- scale WebVR Indoor Scenes. 6.3

[Imae et Hayashibara, 2016] Imae, K. et Hayashibara, N. (2016). ChainVoxel : 3D ChainVoxel : A Data Structure for Scalable Distributed Collaborative Editing for 3D Models. 2016(6):1–8. 3.3.4

[Jia *et al.*, 2014] Jia, J., Wang, W. et Hei, X. (2014). An efficient caching algorithm for peer-to-peer 3D streaming in distributed virtual environments. *Journal of Network and Computer Applications*, 42:1–11. 4.3.1

[Jung *et al.*, 2012] Jung, Y., Behr, J., Drevensek, T. et Wagner, S. (2012). Declarative 3D approaches for distributed web-based scientific visualization services. *CEUR Workshop Proceedings*, 869. 2.1.2

[Khronos, 2007] Khronos (2007). OpenGL ES 2. 2.1.2

[Khronos, 2008] Khronos (2008). OpenGL ES 3. 2.1.2

[Khronos, 2011] Khronos (2011). WebGL 1.0. 2.1.2

[Khronos, 2016] Khronos (2016). WebGL 2.0. 2.1.2

[Klamer, 2013] Klamer, J. (2013). *Conflict resolution in an event sourcing environment*. Thèse de doctorat. 2.3.5

[Koskela *et al.*, 2015] Koskela, T., Heikkinen, A., Harjula, E., Levanto, M. et Ylianttila, M. (2015). RADE : Resource-aware Distributed Browser-to- browser 3D Graphics Delivery in the Web. *IEEE Wireless and mobile*, pages 500–508. 3.2

[Koskela *et al.*, 2014] Koskela, T., Vatjus-anttila, J. et Dahl, T. (2014). Communication Architecture for a P2P-enhanced Virtual Environment Client in a Web Browser. pages 1–5. 2.2.2, 2.2.3

[Kosmadoudi *et al.*, 2013] Kosmadoudi, Z., Lim, T., Ritchie, J., Louchart, S., Liu, Y. et Sung, R. (2013). Engineering design using game-enhanced CAD : The potential to augment the user experience with game elements. *CAD Computer Aided Design*, 45(3). 1.1.2

[Kounev *et al.*, 2008] Kounev, S., Bacon, J., Sachs, K. et Buchmann, A. (2008). A methodology for performance modeling of distributed event-based systems. *Proceedings - 11th IEEE Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, ISORC 2008*, pages 13–22. 2.3.2

[Lacoche *et al.*, 2014] Lacoche, J., Duval, T., Arnaldi, B., Maisel, E. et Royan, J. (2014). A survey of plasticity in 3D user interfaces. *7th Workshop on Software Engineering and Architectures for Realtime Interactive Systems*. 6.3

[Lamport, 1978] Lamport, L. (1978). Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(7):558–565. 3.2.5

[Lavoué *et al.*, 2013] Lavoué, G., Chevalier, L. et Dupont, F. (2013). Streaming compressed 3D data on the web using JavaScript and WebGL. *Proceedings of the 18th International Conference on 3D Web Technology - Web3D '13*, page 19. 4.3.1

[Lewis, 1995] Lewis, J. (1995). IBM Computer Usability Satisfaction Questionnaires : Psychometric Evaluation and Instructions for Use. *International Journal of Human-Computer Interaction*, 7(1):57–78. 5.3.1

[Lewis, 1993] Lewis, J. R. (1993). Multipoint scales : Mean and median differences and observed significance levels. *International Journal of Human-Computer Interaction*, 5(4): 383–392. 5.3.1

[Li *et al.*, 2015] Li, J., Chou, J.-K. et Ma, K.-L. (2015). High performance heterogeneous computing for collaborative visual analysis. *SIGGRAPH Asia 2015 Visualization in High Performance Computing on - SA '15*, pages 1–4. 2.2.2, 2.3.2, 6.3

[Limper *et al.*, 2014] Limper, M., Thöner, M., Behr, J. et Fellner, D. W. (2014). SRC - a streamable format for generalized web-based 3D data transmission. *Proceedings of the Nineteenth International ACM Conference on 3D Web Technologies - Web3D '14*, pages 35–43. 4.3.1

[Lowet et Goergen, 2009] Lowet, D. et Goergen, D. (2009). Co-Browsing Dynamic Web Pages. *Proceedings of the 18th International Conference on World Wide Web - WWW '09*, pages 941–950. 2.1.2

[Lu *et al.*, 2016] Lu, Z., Guerrero, P., Mitra, N. J. et Steed, A. (2016). Open3D : Crowd-Sourced Distributed Curation of City Models. *Web3D '16 : Proceedings of the 21th International Conference on 3D Web Technology*, pages 87–94. 2.1.2

[Maglo *et al.*, 2013] Maglo, A., Lavoue, G., Dupont, F. et Hudelot, C. (2013). 3D mesh compression : survey , comparisons and emerging trends. *ACM Computing Surveys*, 9(4). 4.3.1

[Martinez G. *et al.*, 2009] Martinez G., A., Orozco, H., Ramos, F. et Siller, M. (2009). A Peer-to-Peer Architecture for Real-Time Distributed Visualization of 3D Collaborative Virtual Environments. *2009 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*. 2.2.3

[Méric de Bellefon, 2012] Méric de Bellefon, S. (2012). Une base de données purement fonctionnelle | OCTO talks ! 3.3.4

[Morin, 1990a] Morin, E. (1990a). *Introduction à la pensée complexe*. ESF Editeur. 1

[Morin, 1990b] Morin, E. (1990b). Science avec conscience. 1.1.1

[Mouton *et al.*, 2011] Mouton, C., Grimstead, I. et Cardiff, U. (2011). Collaborative Visualization Current Systems and Future Trends. *Proceedings of the 16th International Conference on 3D Web Technology*, 1:101–110. 2.1.2

[Mouton *et al.*, 2014] Mouton, C., Parfouru, S., Jeulin, C., Dutertre, C., Goblet, J.-L., Paviot, T., Lamouri, S., Limper, M., Stein, C., Behr, J. et Jung, Y. (2014). Enhancing the Plant Layout Design Process using X3DOM and a Scalable Web3D Service Architecture. 2.2

[Nielsen, 1993] Nielsen, J. (1993). *Usability Engineering*, volume 1. AP Professional. 2.2

[Ogden *et al.*, 2017] Ogden, M., Mckelvey, K. et Madsen, M. B. (2017). Dat - Distributed Dataset Synchronization And Versioning. (May). 2.1.1

[Oki *et al.*, 1993] Oki, B., Pfluegl, M., Siegel, A. et Skeen, D. (1993). The Information Bus. *Proceedings of the fourteenth ACM symposium on Operating systems principles - SOSP '93*, pages 58–68. 2.3.2

[Pacull *et al.*, 1994] Pacull, F., Sandoz, A. et Schiper, A. (1994). Duplex : A Distributed Collaborative Editing Environment in Large Scale. *Proceedings of the 1994 ACM conference on Computer supported cooperative work - CSCW '94*, pages 165–173. 2.1.1

[Papageorgiou *et al.*, 2011] Papageorgiou, N., Verginadis, Y., Apostolou, D. et Mentzas, G. (2011). Collaboration pattern assistant. *Proceedings of the 5th ACM international conference on Distributed event-based system - DEBS '11*, page 387. 2.3.1

[Parzyjegla *et al.*, 2010] Parzyjegla, H., Graff, D., Schröter, A., Richling, J. et Mühl, G. (2010). Design and implementation of the Rebeca publish/subscribe middleware. *In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6462 LNCS, pages 124–140. Springer, Berlin, Heidelberg. 2.3.2

[Pietzuch et Bacon, 2002] Pietzuch, P. et Bacon, J. (2002). Hermes : a distributed event-based middleware architecture. *22nd International Conference on Distributed Computing Systems Workshops (ICDCS 2002)*, pages 611–618. 2.3.2

[Ponchio et Dellepiane, 2015] Ponchio, F. et Dellepiane, M. (2015). Fast decompression for web-based view-dependent 3D rendering. *In Web3D '15 : Proceedings of the 20th International Conference on 3D Web Technology*, pages 199–207. 4.3.1

[Prakash et Knister, 1994] Prakash, A. et Knister, M. J. (1994). A framework for undoing actions in collaborative systems. *ACM Transactions on Computer-Human Interaction*, 1(4):295–330. 2.1.1, 2.3.5

[Ressel *et al.*, 1996] Ressel, M., Nitsche-Ruhland, D. et Gunzenhäuser, R. (1996). An integrating, transformation-oriented approach to concurrency control and undo in group editors. *Proceedings of the 1996 ACM conference on Computer supported cooperative work CSCW 96*, pages 288–297. 2.1.1

[Ripeanu,] Ripeanu, M. Peer-to-peer architecture case study : Gnutella network. *In Proceedings First International Conference on Peer-to-Peer Computing*, pages 99–100. IEEE Comput. Soc. 2.3.2

[Roberto *et al.*, 2014] Roberto, D., Dias, C., Durelli, R. S., Remo, J., Brega, F., Gnecco, B. B., Trevelin, L. C. et Guimarães, M. D. P. (2014). Data Network in Development of 3D Collaborative Virtual Environments : A Systematic Review. *LNCS*, 8579:769–785. 2.2.3

[Salvati *et al.*, 2015] Salvati, G., Santoni, C., Tibaldo, V. et Pellacini, F. (2015). MeshHisto. *ACM Transactions on Graphics*, 34(6):1–10. 3.3.3

[Shapiro *et al.*, 2011] Shapiro, M., Pregui, N., Baquero, C. et Zawirski, M. (2011). Replicated Data Types To cite this version : A comprehensive study of Convergent and Commutative Replicated Data Types. (RR-7506):50. 3.3.4

[Shapiro et Preguiça, 2007] Shapiro, M. et Preguiça, N. (2007). Designing a commutative replicated data type. *arXiv preprint arXiv :0710.1784*. 2.1.1

[Singhal et Zyda, 1999] Singhal, S. et Zyda, M. (1999). Networked Virtual Environments. (April):222–226. 1.1.2

[Sons *et al.*, 2010] Sons, K., Klein, F., Rubinstein, D., Byelozyorov, S. et Slusallek, P. (2010). XML3D – Interactive 3D Graphics for the Web. *Proceedings of the 15th International Conference on Web 3D Technology*, pages 175–184. 2.1.2

[Steiakaki *et al.*, 2016] Steiakaki, M., Kontakis, K. et Malamos, A. G. (2016). Real-Time Collaborative environment for interior design based on Semantics , Web3D and WebRTC. *International Symposium on Ambient Intelligence and Embedded Systems*, pages 22–24. 2.2.2

[Stein *et al.*, 2014] Stein, C., Limper, M. et Kuijper, A. (2014). Spatial data structures for accelerated 3D visibility computation to enable large model visualization on the web. *Proceedings of the Nineteenth International ACM Conference on 3D Web Technologies - Web3D '14*, pages 53–61. 2.1.2

[Steinfield *et al.*, 1999] Steinfield, C., Jang, C.-Y. et Pfaff, B. (1999). Supporting Virtual Team Collaboration - TeamSCOPE System. *International Conference on Supporting Group Work (GROUP'99)*, pages 81–90. 1.1.4

[Sun, 2002] Sun, C. (2002). Undo as concurrent inverse in group editors. *ACM Transactions on Computer-Human Interaction*, 9(4):309–361. 2.1.1, 2.3.5

[Sun *et al.*, 1998] Sun, C., Jia, X., Zhang, Y., Yang, Y. et Chen, D. (1998). Achieving Convergence , Causality Preservation , and Intention Preservation in Real-Time Cooperative Editing Systems. *ACM Trans. Comput.-Hum. Interact.*, 5(1):63–108. 1.1.3, 3.3

[Sun *et al.*, 1997] Sun, C., Zhang, Y., Jia, X. et Yang, Y. (1997). A generic operation transformation scheme for consistency maintenance in real-time cooperative editing systems. *Proceedings of the international ACM ...*, pages 425–434. 1.1.3

[Sung *et al.*, 2006] Sung, M. Y., Yoo, Y., Jun, K., Kim, N. J. et Chae, J. (2006). Experiments for a collaborative haptic virtual reality. *Proceedings - 16th International Conference on Artificial Reality and Telexistence - Workshops, ICAT 2006*, pages 174–179. 2.2.3

[Sutter, 2015] Sutter, J. (2015). A CSS Integration Model for Declarative 3D. *In Web3D '15 : Proceedings of the 20th International Conference on 3D Web Technology*, pages 209–217. 2.1.2

[Tarkoma, 2012] Tarkoma, S. (2012). Research Solutions. *In Publish/Subscribe Systems*, pages 205–237. John Wiley & Sons, Ltd, Chichester, UK. 2.3.2

[Tominski, 2006] Tominski, C. (2006). Event-Based Visualization for User-Centered Visual Analysis. *Computer*. 3.2.1

[Verginadis *et al.*, 2009] Verginadis, Y., Apostolou, D., Papageorgiou, N. et Mentzas, G. (2009). Collaboration Patterns in Event-Driven Environments for Virtual Organizations. Rapport technique. 2.3.1

[Vernon, 2013] Vernon, V. (2013). *Implementing Domain-Driven Design*. Addison-Wesley Longman Publishing Co., Inc. 2.3.3

[Vidot, 2002] Vidot, N. (2002). *Convergence des copies dans les environnements collaboratifs répartis*. Thèse de doctorat. 1.1.3

[W3C, 2011] W3C (2011). Extensible 3d (X3D). 2.1.2

[Weiss *et al.*, 2009] Weiss, S., Urso, P. et Molli, P. (2009). An Undo Framework for P2P Collaborative Editing. *In* Bertino, E. et Joshi, J. B. D., éditeurs : *Collaborative Computing : Networking, Applications and Worksharing : 4th International Conference, CollaborateCom 2008, Orlando, FL, USA, November 13-16, 2008, Revised Selected Papers*, pages 529–544. 2.1.1, 2.3.5

[Weiss *et al.*, 2010] Weiss, S., Urso, P. et Molli, P. (2010). Logoot-undo : Distributed collaborative editing system on P2P networks. *IEEE Transactions on Parallel and Distributed Systems*, 21(8):1162–1174. 1.1.3, 2.3.5

[Wu *et al.*, 2014] Wu, D., Rosen, D. W. et Schaefer, D. (2014). *Cloud-Based Design and Manufacturing (CBDM)*. 1.1

[Xhafa et Poulovassilis, 2010] Xhafa, F. et Poulovassilis, A. (2010). Requirements for distributed event-based awareness in P2P groupware systems. *24th IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2010*, (October):220–225. 1.1.4

[You et Pekkola, 2001] You, Y. et Pekkola, S. (2001). Meeting others—supporting situation awareness on the WWW. *Decision Support Systems*, 32:71–82. 1.1.4

[Young, 2009] Young, G. (2009). Code Better. 2.3.4

[Young, 2010] Young, G. (2010). CQRS and CAP Theorem. 3.2.5

[Yu *et al.*, 2016] Yu, M., Cai, H., Ma, X. et Jiang, L. (2016). Symmetry-Based Conflict Detection and Resolution Method towards Web3D-based Collaborative Design. *Symmetry*, 8(5):35. (document), 2.1

[Yuan *et al.*, 2002] Yuan, P.-p. Y. P.-p., Chen, G. C. G., Dong, J.-x. D. J.-x. et Han, W.-l. H. W.-l. (2002). Research on an event specification for event-based collaboration support software architecture. *The 7th International Conference on Computer Supported Cooperative Work in Design*, 7:99–104. 2.3.1

[Zhang *et al.*, 2013] Zhang, L., Zhou, F., Mislove, a. et Sundaram, R. (2013). Maygh : Building a CDN from client web browsers. *Proceedings of the 8th ACM European Conference on Computer Systems, EuroSys 2013*, pages 281–294. 2.2.2

[Zhu *et al.*, 2011] Zhu, M., Mondet, S., Morin, G., Ooi, W. T. et Cheng, W. (2011). Towards peer-assisted rendering in networked virtual environments. *Proceedings of the 19th ACM international conference on Multimedia - MM '11*, page 183. 2.2.3, 6.3

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	**Dénomination**	**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 *stream*
CHUNK	Réception d'une donnée *chunk*)
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

Tableau A.3 – Statut du nœud

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 – Modèles utilisés pendant l'expérimentation 1

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

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

Glossaire

Event Store Mémoire permettant le stockage des évènements en mode "*append-only*". 71, 76

Framework (ou structure logicielle en français) est un ensemble de composants génériques proposant un cadre de travail guidant l'architecture logicielle. 13, 35, 49, 60, 76, 85, 90

ICE Framework est une technique dans le domaine du réseau pour trouver le chemin entre deux machine pour communiquer l'une avec l'autre, le plus directement possible en P2P. Le framework permet aux pairs en cours d'établissement de connexion de découvrir et communiquer leur adresse IP publique dans le but de pouvoir être atteint par d'autres pairs.. 33, 82

Snapshot (contexte : CQRS) Instantané de l'état d'un agrégat convertissant l'objet du domaine en un objet de transferts de données (DTO). 45

Streaming 3D est une technique permettant d'envoyer un contenu 3D sous la forme d'un flux continu par le biais d'internet, qui peut être utilisé/lu au fur et à mesure qu'il est reçu. 88

WebSocket Protocole réseau standard du Web visant à créer des canaux de communication full-duplex par dessus une connexion TCP. 20, 30, 80

Acronymes

3D trois dimensions. 2–8, 11–15, 17–25, 27, 30, 33, 34, 38, 40–44, 46, 47, 49–55, 60–63, 66–68, 81, 84, 88–92, 98–104, 107, 110, 126

AIC Architecture, Ingénierie et Construction. 19

API Application Programming Interface. 20, 23, 29–31, 64, 65, 80–82, 86, 87

BIM Business Information Modeling. 3, 4, 6, 11, 19, 47, 98

CAO Conception Assistée par Ordinateur. 2, 6, 11, 15, 19, 20, 40, 47, 98

CAP Consistency, Availability, and Partition Tolerance. 59

CCI Causality, Convergence, Intention. 8, 62

CDN Content Delivery Network. 30

CE Cohérence Éventuelle. 59

CEP Complex Event Processing. 61

CQRS Command Query Responsability Segregation. vi, 35, 41–43, 47–49, 51, 54, 55, 58, 59, 61, 85, 87, 89, 108, 116

CRDT Conflict-Free Replication Data Type. 18, 63

CRUD Create, Read, Update, Delete. 68, 70, 82, 84, 89

CS Command Sourcing. 45, 46

CSCW Computer-Supported Cooperative Work. 34, 50

CSS Cascading Style Sheet. 22

DDD Domain Driven Design. 38–41, 47, 49, 53, 61, 108, 116, 126

DHT Distributed Hash Table. 36, 37

DOM Document Object Model. 22, 23

DTLS Datagram Transport Layer Security. 31

DTO Data Transfer Object. 55

ECMA Ecma International - European association for standardizing information and communication systems. 117

EDA Event Driven Architecture. 9, 12, 37, 47, 51, 52, 98, 103

EP Event Processing. 50, 51

ES Event Sourcing. vi, 35, 40, 42–49, 51, 54, 55, 59, 61, 74, 116, 119, 126

ESM Event Stream Manager. 73, 75, 76

EV Environnement Virtuel. 25, 33

EV 3D Environnement Virtuel 3D. 2

EVC Environnement Virtuel Collaboratif. 20, 24, 25, 33, 34, 47, 50, 53, 115

EVC 3D Environnement Virtuel Collaboratif 3D. 2, 3, 5, 6, 11, 13, 15, 18, 25, 30, 33, 47, 49, 63, 72, 78, 115, 117, 118

HTML HyperText Markup Langage. 22, 23, 90

ICE Interactive Connectivity Establishment. 33

IETF Internet Engineering Task Force. 29, 117

IHM Interface Homme Machine. 6, 89

IM Instance Manager. 73

IU Interface Utilisateur. 37, 55, 67, 83, 89, 90, 92, 101, 102, 108, 117

JS JavaScript. 80, 82, 83, 117

JSON JavaScript Object Notation. 68, 87

LAN Local Area Network. 26, 33, 100

MAN Metropolitan Area Network. 26

NAT Network Address Translator. 32, 33, 82

NB Network Bridge. 73, 74

ORM Object-Relational Mapping. 74

OT Operational Transformation. 17, 18

OWL Web Ontology Language. 35

P2P pair à pair. 10, 14, 19, 20, 29–31, 34, 36, 37, 47, 49, 51, 58, 62–70, 73, 78, 80, 82, 83, 86, 88, 102, 115, 117, 118, 120

PDM Product Data Management. 19

PLM Product Lifecycle Management. 6, 19

Pub / Sub Publish-Subscribe. 11, 35–37, 47, 83

REST REpresentational State Transfer. 67, 68, 82

RRTCC Receiver-side Real-Time Congestion Control. 31

RTCP Real-time Transfert Control Protocol. 25, 26, 33

RTP Real-Time Transport Protocol. 25, 26, 33

RV Réalité Virtuelle. 20

SCTP Stream Control Transmission Protocol. 28, 29, 31, 33, 34

SEC Système d'Edition Collaborative. 27, 62

SIG Système d'Information Géographique. 22

STUN Simple Traversal of UDP through NATs. 32, 33

SVG Scalable Vector Graphics. 22

TCP Transmission Control Protocol. 28–30, 33

TURN Traversal Using Relays around NAT. 33

UDP User Datagram Protocol. 28–30, 33

W3C World Wide Web Consortium. 15, 29, 46, 117

WAN Wide Area Network. 26, 33

WebRTC Web Real-Time Communication. 29, 30, 51, 80, 82, 86, 92

Résumé

L'évolution technologique du web durant ces dernières années a favorisé l'arrivée d'environnements virtuels collaboratifs pour la modélisation 3D à grande échelle. Alors que la collaboration réunit dans un même espace partagé des utilisateurs distants géographiquement pour un objectif de collaboration commun, les ressources matérielles qu'ils apportent (calcul, stockage, 3D ...) avec leurs connaissances sont encore trop rarement utilisées et cela constitue un défi. Il s'agit en effet de proposer un système simple, performant et transparent pour les utilisateurs afin de permettre une collaboration efficace à la fois sur le volet computationnel mais aussi, bien entendu, sur l'aspect métier lié à la modélisation 3D sur le web. Pour rendre efficace le passage à l'échelle, de nombreux systèmes utilisent une architecture réseau dite "hybride", combinant client serveur et pair-à-pair. La réplication optimiste s'adapte bien aux propriétés des ces environnements répartis : la dynamicité des utilisateurs et leur nombre, le type de donnée traitées (3D) et leur taille.

Cette thèse présente un modèle pour les systèmes d'édition collaborative en 3D sur le web. L'architecture cliente (3DEvent) permet de déporter les aspects métiers de la 3D au plus près de l'utilisateur sous la forme d'évènements. Cette architecture orientée événements repose sur le constat d'un fort besoin de traçabilité et d'historique sur les données 3D lors de l'assemblage d'un modèle. Cet aspect est porté intrinsèquement par le patron de conception event-sourcing. Ce modèle est complété par la définition d'un intergiciel en pair-à-pair. Sur ce dernier point, nous proposons d'utiliser la technologie WebRTC qui présente une API familière aux développeurs de services en infonuagique. Une évaluation portant sur deux études utilisateur concernant l'acceptance du modèle proposé a été menée dans le cadre de tâches d'assemblage de modèles 3D sur plusieurs groupes d'utilisateurs.

Mot clés : environnement virtuel collaboratif, réseau pair-à-pair, WebRTC, web 3D, conception 3D, traitement réparti évènementiel, architecture hybride, event-sourcing.

Abstract

Web technologies evolutions during last decades fostered the development of collaborative virtual environments for 3D design at large scale. Despite the fact that collaborative environments gather in a same shared space geographically distant users in a common objective, the hardware ressources of their clients (calcul, storage, graphics ...) are often underused because of the challenge it represents. It is indeed a matter of offering an easy-to-use, efficient and transparent collaborative system to the user supporting both computationnal and 3D design visualisation and business logic needs in heterogeneous web environments. To scale well, numerous systems use a network architecture called "hybrid", combining both client-server and peer-to-peer. Optimistic replication is well adapted to distributed application such as 3D collaborative envionments : the dynamicity of users and their numbers, the 3D data type used and the large amount and size of it.

This document presents a model for 3D web-based collaborative editing systems. This model integrates 3DEvent, an client-based architecture allowing us to bring 3D business logic closer to the user using events. Indeed, the need of traceability and history awareness is required during 3D design especially when several experts are involved during the process. This aspect is intrinsec to event-sourcing design pattern. This architecture is completed by a peer-to-peer middleware responsible for the synchronisation and the consistency of the system. To implement it, we propose to use the recent web standard API called WebRTC, close to cloud development services know by developers. To evaluate the model, two user studies were conducted on several group of users concerning its responsiveness and the acceptance by users in the frame of cooperative assembly tasks of 3D models.

Keywords : collaborative virtual environment, peer-to-peer network, WebRTC, Web 3D, 3D design, distributed event-based system, hybrid architecture, event-sourcing.