

Architecture évènementielle pour les
Environnements Virtuels Collaboratifs sur le web :
Application à la manipulation et à la visualisation
d'objets 3D

Caroline Desprat

1^{er} octobre 2017

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 réuni 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 et la visualisation 3D dans un environnement hétérogènes en termes de performances de calcul, de rendu et de connexion. Pour rendre efficace le passage à l'échelle, en conservant une source de vérité centralisée, de nombreux systèmes utilisent une architecture réseau dite "hybride", combinant client serveur et pair-à-pair. Cependant, la synchronicité élevée et la réPLICATION des données sur tous les sites peut mener à une divergence des copies dans un environnement réparti. C'est pourquoi il est nécessaire de s'intéresser à la réPLICATION optimiste adaptée aux propriétés ces environnements collaboratifs 3D : la dynamicité des utilisateurs et leur nombre, le type de donnée traitées (3D) et leur masse. Le cadre d'un système collaboratif impose également la conservation des propriétés de Causalité, Convergence et préservation de l'Intention (CCI).

Cette thèse présente un modèle pour les systèmes d'édition collaborative en 3D dans un environnement web. Dans ce modèle est intégrée une architecture cliente (3DEvent) qui permet de déporter les aspects métiers de la 3D au plus près de l'utilisateur sous la forme d'évènements. La mise en place de cette architecture basée-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 une technologie récente comme 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 environments in terms of computing, rendering and connexion performances. To scale well while conserving a centralised authoritative source of data, numerous systems use a network architecture called "hybrid", combining both client-server and peer-to-peer. However, real-time updates and data replication on different sites lead to divergence of copy in such a distributed environment. That is why optimistic replication is well adapted to 3D collaborative envionments by taking into account different parameters : the dynamicity of users and their numbers, the 3D data type used and the large amount and size of it. It is also imperative to respect collaborative system properties of Causality, Convergence and Intention preservation (CCI).

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 traçability 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.

Table des matières

Table des figures	ix
Liste des tableaux	xi
Liste des abréviations	xiii
1 Introduction	1
1.1 Contexte	2
1.1.1 La collaboration	2
1.1.2 Les environnements virtuels collaboratifs 3D	6
1.1.3 Les systèmes d'édition collaboratifs	7
1.1.4 Les architectures orientées évènements 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	15
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 et au delà	20
2.1.3 Web 3D : deux approches	23
2.2 Communication en temps-réel	25
2.2.1 Fiable versus Non Fiable	27
2.2.2 Ordonné versus Désordonné	28
2.2.3 Les principaux protocoles de transport	29
2.2.4 Web et P2P : WebRTC	31
2.2.5 Quel protocole dans quel EVC3D ?	35
2.3 Les systèmes distribués orientés évènements pour la collaboration	36
2.3.1 Introduction	36
2.3.2 Systèmes Publish-Subscribe	37
2.3.3 Domain Driven Design	40
2.3.4 Command Query Responsability Segregation	43
2.3.5 Event Sourcing	44
2.4 Conclusion	49

3 Contributions scientifiques	51
3.1 Introduction	52
3.2 Modèle évènementiel 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 Responsibility Segregation (CQRS)	58
3.2.3 Cohérence Éventuelle en CQRS	61
3.2.4 Potentielles applications et autres utilisations	63
3.2.5 Bilan	64
3.3 Architecture de communication hybride	64
3.3.1 Présentation générale	66
3.3.2 Extension : Event Store distribué	68
3.3.3 Persistance à long terme	69
3.3.4 Synchronisation client-serveur	69
3.3.5 Mécanisme de gestion de version	70
3.3.6 Gestion de la cohérence	71
3.3.7 Bilan	71
3.4 Conclusion du chapitre	71
4 Implantation	73
4.1 Introduction	74
4.2 3DEvent : Plateforme web de manipulation collaborative d'objets 3D	74
4.2.1 Éditeur 3DEvent	74
4.2.2 Interface utilisateur	75
4.2.3 Flexibilité de la visualisation	78
4.3 Intergiciel P2P pour l'échange de données 3D	78
4.3.1 Données d'échange	80
4.3.2 Synchronisation des données	81
4.4 Résumé des choix techniques	83
4.5 Bilan	84
5 Expérimentations	85
5.1 Introduction	86
5.2 Cas d'étude : Assemblage collaboratif d'objets 3D dans un environnement web	86
5.3 Expérimentation 1 : preuve de faisabilité	86
5.3.1 Présentation de l'expérimentation	86
5.3.2 Résultats	86
5.3.3 Discussion et Conclusion	86
5.4 Expérimentation 2 : Intégration du framework évènementiel	86
5.4.1 Résultats et Discussion	89
5.5 Comparaison entre l'expérimentation 1 et l'expérimentation 2	91
5.5.1 Résultats	91
5.5.2 Discussion et Conclusion	91
5.6 Bilan	91
6 Conclusion	97

A Ressources pour l'implantation et les expérimentations	99
A.1 Description des évènements par agrégat dans 3DEvent	99
A.2 Messages réseaux pour la synchronisation des Event Stores	101
A.3 Expérimentation 2 : User study questionnaire	102
Bibliographie	103

Table des matières

Table des figures

1.1	Place de la contribution dans les différents champs de recherche	14
2.1	Déclaratif vs Impératif en 2D et en 3D sur le web	23
2.2	Support de WebGL 1 (2014-2017) et WebGL 2 (2016-2017)	24
2.3	Pile des protocoles IP: TCP vs UDP	33
2.4	STUN, TURN et signalisation	34
2.5	Architecture Publish-Subscribe	38
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)	44
2.8	Transaction en ES	46
2.9	Snapshot en ES	46
3.1	Illustration des notions de évènements du domaine, des types d'évènements, et des instances d'évènements.	56
3.2	Modèle de l'architecture client dans 3DEvent	58
3.3	Théorème CAP et les algorithmes de compromis	62
3.4	Composants de chaque pair dans 3DEvent (point de vue réseau)	67
3.5	Protocole de connexion au réseau d'instance 3DEvent	68
4.1	Flux de la collaboration dans le framework 3DEvent entre 3 utilisateurs	77
4.2	Structure d'un agrégat	79
5.1	Interface utilisateur pendant une session collaborative (trois personnes)	92
5.2	Persistance long-terme (Event Store [®]), base de données/outil de monitoring	93
5.3	Résumé d'une session collaborative au cours du temps	94
5.4	Résultats des questionnaires collectés	95

Table des figures

Liste des tableaux

2.1	Solutions conventionnelles pour les problèmes collaboratifs [103].	18
2.2	Encodage des données transmises	22
2.3	Aperçu des procotoles de transport	31
3.1	Évènements de pour l'agrégat Maillage (extrait de Tab. A.1)	57
5.1	Modèles utilisés durant l'expérimentation	87
A.1	3DEvent : résumé des évènements par agrégats	100
A.2	Type de messages lors de la synchronisation	101
A.3	Statut du noeud	101

Liste des tableaux

Liste des abréviations

- 3D** tridimension. 2, 12–15, 60
- AIC** Architecture, Ingénierie et Construction. 19
- API** Application Programming Interface. 19, 22, 28, 29, 59, 60, 69, 71
- BIM** Business Information Modeling. 3, 4, 6, 12, 19, 45
- CAO** Conception Assistée par Ordinateur. 2, 6, 12, 15, 19, 20, 38, 45
- CAP** Consistency, Availability, and Partition Tolerance. 56
- CCI** Causality, Convergence, Intention. 7, 59
- CDN** Content Delivery Network. 28
- CE** cohérence éventuelle. 55
- CEP** Complex Event Processing. 57
- cloud** Définir le cloud. 49
- CQRS** Command Query Responsibility Segregation. 33, 39–41, 44, 47, 49, 52, 53, 56, 58, 68, 83
- CRDT** Conflict-Free Replication Data Type. 18
- CS** Command Sourcing. 43, 44
- CSCW** Computer-Supported Cooperative Work. 33, 48
- CSS** Cascading Style Sheet. 22
- DDD** Domain Driven Design. 36–39, 44, 47, 50, 58
- DHT** Distributed Hash Table. 34, 35
- DOM** Document Object Model. 21, 22
- DTLS** Datagram Transport Layer Security. 29
- DTO** Data Transfer Object. 52
- EP** event processing. 48
- ES** Event Sourcing. 33, 38, 40–44, 47, 49, 52, 53, 56–58, 83
- EV** Environnement Virtuel. 23, 31
- EVC** Environnement Virtuel Collaboratif. 20, 23, 24, 32, 48, 51, 60, 84
- EVC 3D** Environnement Virtuel Collaboratif 3D. 2, 3, 5, 6, 11, 13, 15, 18, 23, 24, 28, 31, 32, 47, 64, 84

event store Mémoire permettant le stockage des évènements en mode "*append-only*".
[44](#), [63](#)

Framework (ou structure logicielle en français) est un ensemble de composants génériques proposant un cadre de travail guidant l'architecture logicielle.
[13](#), [33](#), [57](#), [64](#), [66](#), [67](#)

glTF GL Transmission Format.
[71](#)

HTML HyperText Markup Langage.
[22](#), [68](#)

ICE Interactive Connectivity Establishment.
[30](#)

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..
[30](#)

IETF Internet Engineering Task Force.
[28](#)

IU Interface Utilisateur.
[36](#), [52](#), [67](#)

JSON JavaScript Object Notation.
[71](#)

LAN Local Area Network.
[25](#), [31](#), [32](#)

MAN Metropolitan Area Network.
[25](#)

NAT Network Address Translator.
[30](#)

NoSQL Not Only SQL.
[72](#)

OT Operational Transformation.
[17](#), [18](#)

OWL Web Ontology Language.
[33](#)

P2P pair à pair.
[9](#), [10](#), [14](#), [18](#), [19](#), [28](#), [29](#), [32](#), [34](#), [35](#), [45](#), [49](#), [59–61](#), [68](#)

PDM Product Data Management.
[19](#)

PLM Product Lifecycle Management.
[6](#), [19](#)

PubSub Publish-Subscribe.
[10](#), [33–35](#), [44](#)

RRTCC Receiver-side Real-Time Congestion Control.
[29](#)

RTCP Real-time Transfert Control Protocol.
[24](#), [25](#), [32](#)

RTP Real-Time Transport Protocol.
[24](#), [25](#), [32](#)

RV Réalité Virtuelle.
[19](#)

SCTP Stream Control Transmission Protocol.
[27](#), [29](#), [32](#)

SEC Système d'Edition Collaborative.
[26](#), [58](#), [59](#)

SIG Système d'Information Géographique.
[22](#)

- Snapshot** (contexte : CQRS) Instantané de l'état d'un agrégat convertissant l'objet du domaine en un objet de transferts de données (DTO). [43](#)
- 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. [72](#)
- STUN** Simple Traversal of UDP through NATs. [30](#), [31](#)
- SVG** Scalable Vector Graphics. [21](#)
- TCP** Transmission Control Protocol. [26–28](#), [32](#), [60](#)
- TURN** Traversal Using Relays around NAT. [30](#), [31](#)
- UDP** User Datagram Protocol. [26–28](#), [31](#), [32](#), [60](#)
- W3C** World Wide Web Consortium. [15](#), [28](#)
- WAN** Wide Area Network. [25](#), [31](#)
- WebRTC** Web Real-Time Communication. [28](#), [49](#), [59](#), [60](#), [70](#)
- WebSocket** Protocole réseau standard du Web visant à créer des canaux de communication full-duplex par dessus une connexion TCP. [20](#), [28](#), [60](#), [70](#)

Liste des abréviations

Chapitre 1

Introduction

Contents

1.1	Contexte	2
1.1.1	La collaboration	2
1.1.2	Les environnements virtuels collaboratifs 3D	6
1.1.3	Les systèmes d'édition collaboratifs	7
1.1.4	Les architectures orientées évènements 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	15

1.1 Contexte

Le besoin de collaboration et de manipulation en temps réel d'objets en **tri-dimension (3D)** ainsi que leur contrôle de version a été très tôt une raison de la dissémination des outils pour la **Conception Assistée par Ordinateur (CAO)**. Tandis que l'ingénierie et la visualisation scientifique ont produit des quantités de données massives dans des domaines tels que la conception architecturale, l'industrie des jeux ou encore l'impression **3D**, un besoin croissant de maintenir, visualiser et manipuler de larges scènes **3D** polygonales pouvant être éditées par de multiples utilisateurs de manière concurrente se fait sentir [16, 98]. 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 de répondre à ces besoins est en forte progression. Dans un **Environnement Virtuel 3D (EV 3D)**, la simulation virtuelle et simulation distribuée d'un environnement 3D, un grand nombre d'utilisateurs situés à différents endroits géographiques peuvent interagir les uns avec les autres en temps réel. Un **Environnement Virtuel Collaboratif 3D (EVC 3D)** insiste sur l'aspect collaboratif des interactions qui se produisent entre les utilisateurs. Cette distorsion de l'espace physique et temporel impose aux **EVCs 3D** des mécanismes de communication rapides, conservant un environnement cohérent des données partagées durant la collaboration. L'utilisation d'un client comme medium pour accéder à l'**EVC 3D** est requise pour envoyer des demandes au serveur. Le client peut être manipulé par un utilisateur ou un agent logiciel (bot). Un **EVC 3D** se compose également d'un protocole de communication qui permet aux différents clients d'échanger les mises à jours correspondants aux modifications effectuées dans l'espace 3D virtuel partagé. La distribution des données devient alors un enjeu majeur dans ce type d'application en terme de temps, de sécurité et de fiabilité.

1.1.1 La collaboration

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

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 » [65, p. 100].

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 transdisciplinarité sont acceptés comme dans le concept de pensée complexe. Dans sa définition de la complexité, Edgar Morin fait d'ailleurs référence à l'étymologie latine « *complexus* » qui signifie « ce qui est tissé ensemble » [66]. La plupart des collaborations requièrent un élément dirigeant qui peut prendre une forme sociale (personne) au sein d'un groupe décentralisé et égalitaire (tout le monde au même niveau). L'élément dirigeant va également souvent aider à trouver des consensus. La disponibilité des ressources peut aussi devenir un élément dirigeant dans la collaboration. Une équipe travaillant de manière collaborative a la possibilité de concentrer plus de ressources, de reconnaissances et de récompenses lors d'une compétition comportant des ressources finies. La collaboration est aussi présente dans la recherche de buts opposés mettant en avant la notion de collaboration contradictoire (en opposition avec la collaboration constructive) ; la négociation et la compétition peuvent également faire partie du terrain de la collaboration.

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

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

Dans les années 2000, deux classifications ont été retenues concernant la collaboration. La première classification, décrite en 2007 par Gotta [33], propose un modèle segmentant la collaboration de manière structurée en quatre catégories : de la plus dirigée à la plus volontaire, en passant par l'hybride.

- *Collaboration centrée processus.* Les conditions requises du processus nécessitent l'engagement de l'utilisateur, qui doit, de par son rôle ou sa responsabilité, diriger ses efforts dans la collaboration avec les autres. Cette stratégie se concentre sur les activités de manipulation collaborative 3D plutôt que sur leur contexte organisationnel afin de favoriser la synergie autour de la réussite d'un processus. Par exemple, pour la création et l'utilisation d'un modèle 3D dans un **Business Information Modeling (BIM)**, il s'agit de favoriser la prise de décision sur un projet et de communiquer à ce propos.

- *Collaboration centrée activité.* Les activités partagées créent un sentiment de co-dépendance qui induit la collaboration entre les membres. La co-dépendance prend l'avantage sur l'intérêt personnel comme motivation à collaborer. Le groupe a besoin de chacun pour que l'objectif soit considéré comme réalisé. L'intérêt personnel ou l'allégeance à l'esprit d'équipe peut aussi promouvoir la collaboration. Par exemple, la visualisation collaborative des activités des différents contributeurs dans l'**EVC 3D** permet à chacun de rendre compte de ses réalisations. Ce type de collaboration doit beaucoup à l'ergonomie de l'activité qui insiste sur la différence entre le travail prescrit et le travail réel : la tâche et l'activité.
- *Collaboration centrée communauté.* La participation de la communauté à la collaboration induit la contribution. En effet, les interactions professionnelles ou sociales peuvent encourager ou persuader les utilisateurs de partager leurs informations ou connaissances (exemple : les logiciels *open-source*)
- *Collaboration centrée réseau.* Les connexions réseau favorisent la coopération réciproque. Dans le but de récupérer des avis ou du savoir-faire externe, un utilisateur peut faire appel à son réseau social pour compléter une autre interaction collaborative. Souvent utilisée dans le cadre d'urgences écologiques ou sanitaires (exemple : contributions OpenStreetMap lors d'ouragans ou de tsunamis), la collaboration centrée réseau est très présente dans des situations où l'expertise est fortement valorisée comme dans les **BIM** ou la visualisation scientifique. Les contributeurs peuvent être intégrés en fonction des besoins des utilisateurs déjà présents sur le projet.

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

Une seconde classification proposée par Callahan et al. [12] 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 en réseau.* La collaboration en réseau est une surcouche de la collaboration traditionnellement centrée sur la relation d'une équipe ou d'une communauté. Elle s'appuie sur une action individuelle et un intérêt personnel qui resurgissent ensuite sur le réseau, sous la forme de personnes qui contribuent ou cherchent quelque chose à partir du réseau. L'adhésion et les périodes sont ouvertes et non limitées. Il n'y a pas de rôle explicite. Les membres ne se connaissent pas forcément. Le pouvoir est distribué. Cette forme de collaboration est dirigée par l'avènement des réseaux sociaux, d'un accès omniprésent à internet et de la capacité de se connecter avec divers individus malgré la distance.

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

Le travail dans un **EVC 3D** facilite la compréhension de certaines problématiques liées à l'espace 3D ; c'est également un point de rencontre et d'échanges entre contributeurs sur le court terme et le long terme. Le croisement de ces deux dimensions,

spatiale et temporelle, implique une multiplication des points de vue et donc des données à traiter sur le problème lors de la collaboration.

1.1.2 Les environnements virtuels collaboratifs 3D

Un **EVC 3D** est un environnement virtuel 3D où plusieurs utilisateurs locaux ou à distance peuvent se rejoindre et partager une expérience collaborative interactive en 3D. La principale caractéristique d'un **EVC 3D** est la simulation immersive et interactive 3D d'environnements virtuels, tels que les jeux sérieux ou multijoueurs. **EVC 3D** permet à plusieurs utilisateurs d'interagir les uns avec les autres en temps (quasi) réel, même s'ils sont situés dans des lieux différents. D'autres fonctionnalités sont proposées par ce type de plateformes comme le partage d'espace, de présence et du temps. Selon Singhal et Zyda [80], **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 [38] : 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 lorsqu'on aborde la **CAO**, le **BIM** ou le **Product Lifecycle Management (PLM)**. La manipulation collaborative d'objets virtuels par plusieurs utilisateurs fait ainsi partie des nouveaux besoins liés aux développements de ces activités. Elle s'avère indispensable dans plusieurs types d'applications comme le prototypage virtuel, les simulations d'entraînement ou la simulation d'assemblage et de maintenance. L'expérience de modélisation **CAO** collaborative peut être accompagnée de mécanismes ludiques pour intégrer une capture temps réel de la connaissance [53]. 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.* » [30]. 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, suffisamment 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 [94]. L'aspect temporel se réfère à la synchronicité des interactions. Une synchronicité élevée implique que les actions des utilisateurs sont visibles en temps réel (action synchrone). Au contraire, une synchronicité faible, implique qu'un temps notable s'écoule entre l'action d'un utilisateur et sa visibilité chez les autres (action asynchrone). L'aspect spatial s'intéresse à la répartition géographique des utilisateurs. Les environnements sont dits répartis lorsque les utilisateurs sont physiquement distants. Cette thèse concerne les applications collaboratives appartenant à la catégorie des applications synchrones réparties.

Un modèle d'édition collaborative en temps réel permet à plusieurs utilisateurs de visualiser et d'éditer de manière simultanée le même document (texte, image, 3D) depuis plusieurs sites connectés par un réseau de communication. Un modèle d'édition collaborative se compose d'un nombre connu ou inconnu de répliques. Une réplique est une copie du document partagé modifiable à n'importe quel instant. L'exécution d'une opération de modification se fait localement sur la réplique, qui la diffuse ensuite à l'ensemble des répliques. La notification de la modification qui contient 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 actuel 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 [87]. Le modèle [Causality, Convergence, Intention \(CCI\)](#) proposé par Sun et al. [86] propose un modèle de cohérence pour les systèmes d'édition collaborative qui s'adresse à ces facteurs afin d'assurer « la préservation de la convergence, de la causalité et la préservation de l'intention » [86]. Une majorité de travaux se sont intéressés à ces propriétés sur des documents textuels [97].

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'évènements. Pour cela, différentes descriptions du terme sont proposées ainsi que leur notion liée à la littérature informatique.

Dans la littérature, une variété de définitions du terme évènement existe. Habituellement, un évènement 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 évènements en n'indiquant pas le champ d'application dans lequel ils sont utilisés. Il existe trois entrées pour le terme évènement dans le dictionnaire. La première entrée se réfère à la physique (dans le cadre de la théorie de la relativité), où un évènement 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 évènement est « la partie d'un univers Ω réalisée quand l'une des éventualités la composant se réalise ». La troisième définition se rapporte à la psychologie impliquant « tout ce qui est capable de modifier la réalité interne d'un sujet (fait extérieur, représentation, etc.) ». Cette dernière définition est très proche de ce que l'on retrouve en informatique, comme les changements d'état ou les actions entraînant certaines conséquences. Dans une application, il peut être important de s'intéresser au déplacement d'un objet de quelques unités (*object moved event*) ou d'apprendre qu'un utilisateur est passé de déconnecté à connecté (*status changed*). On s'applique donc à identifier ce qui modifie l'état intrinsèque d'un objet et sa représentation. Bien que ces différentes descriptions permettent d'avoir une compréhension générale du terme évènement, chaque sous-discipline de l'informatique comprend ses propres associations avec le terme évènement.

L'implantation d'une architecture orientée évènements (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 évènements doivent répondre aux exigences des utilisateurs et aux problèmes liés à la nature des plateformes et applications. Le passage à l'échelle (nombre de d'utilisateurs, ressources distribuées sur de grandes zones géographiques) génère un grand nombre d'évènements qui doivent être traités de façon efficace.

Les systèmes complexes distribués sont construits à partir de collections couplées de manière dite « lâche » (*loosely coupled*)², technologiquement neutre et indépen-

2. Un couplage lâche indique que les composants échangent peu d'informations. Contrairement à un couplage fort, où les composants ont une faible indépendance fonctionnelle et sont donc peu

dante de la localisation des services. Le développement d'applications dirigées par les évènements est un challenge tripartite : la production d'évènements, le traitement des évènements et la consommation des évènements [17]. Cristea et al. [22] présentent un aperçu des architectures distribuées pour les systèmes orientés évènements. 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 évènements prenant en compte la sensibilisation au groupe (*awareness*) dans des activités. Ces systèmes sont présents dans la littérature depuis les prémisses de la technologie web [8, 84, 100]. Ces propositions ont commencé par approcher la sensibilisation à l'espace de travail dans le but d'informer les utilisateurs des changements se produisant dans l'espace de travail partagé. Des travaux plus récents se sont concentrés sur des nouveaux paradigmes comme les systèmes **pair à pair (P2P)** pour proposer des *groupware* décentralisés ubiquitaires et sensibles à l'environnement. La sensibilisation au sein du groupe n'est pas seulement désignée 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 et la fourniture d'un système de communication ; la disponibilité d'informations superflues ; la difficulté d'extraire l'information pour surveiller ou faire des rapports ; l'utilisation des données issues des ressources produites par le groupe pour améliorer sa perception des données disponibles. Par exemple, Xhafa et Poulovassilis [99] exposent une approche distribuée basée évènements pour gérer des collectifs (*groupware*) en **P2P**. Leur méthode permet de développer des applications de collectifs spécifiques sur les opérations et les services primitifs liés à la sensibilisation. 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 A l'inverse d'une approche centralisée, où l'information de sensibilisation est gérée par un serveur central, dans un système

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.

décentralisé, l'information sur laquelle la sensibilisation est construite est répartie sur les sites des différents pairs. Dans les approches basées serveur, le traitement des événements est effectué sur le serveur qui stocke les événements ainsi que la base de données contenant l'historique et fournit l'interface de requête pour extraire les informations d'intérêt. Dans le collecticiel P2P, le stockage, le traitement et les requêtes d'événements doivent être faits de manière répartie.

La sensibilisation à la dynamicité des événements Les systèmes P2P sont dynamiques par nature (*join / leave* des pairs). La fonctionnalité de sensibilisation doit prendre en compte cette dynamicité. En effet, la synchronisation et la cohérence de 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 évènements manipule plusieurs types d'évènements. C'est pourquoi il est nécessaire que les évènements soient le plus génériques possibles pour faciliter la construction de requêtes à travers un système lâchement couplé.

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

Brown et al. [10] proposent un mécanisme de distribution de données basé évènements. 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 évènements non voulus. L'analyse des applications utilisant des systèmes orientés évènements explique qu'ils sont obligés

de faire des choix spécifiques au métier lié à l’application [40]. Par exemple, dans les applications de jeux vidéo, la distribution des évènements est souvent liée à un mécanisme **Publish-Subscribe (PubSub)** centralisé qui modifie les souscriptions au fur et à mesure que les joueurs se déplacent dans l'espace. De plus, les évènements dans un jeu doivent être protégés contre l'altération pour éviter la triche ou la dissémination d'évènements 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 évènements sont bien adaptées pour intégrer dans la description des évènements plusieurs aspects liés au métier. Cette sémantique porte avec elle les connaissances des manipulations expertes. Un aspect majeur de cette thèse repose sur l'intégration du métier dans le processus de la manipulation et de la visualisation des objets **3D**.

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

1.2 Problématique

Cette thèse se situe à l'interface de trois champs de recherches (Figure 1.1). Le premier concerne les environnements de modélisation **3D**. Souvent commerciaux (Clara.io, OnShape, Verold Studio), les modeleurs utilisent des technologies supportées

par les navigateurs web qui leur permettent d'être disponibles sur la plupart des plateformes. Cependant ces dernières reposent sur une gestion centralisée des données qui rend les utilisateurs très dépendants de la disponibilité de ces plateformes et d'une connexion internet pour la distribution des informations. Mises à part quelques exceptions, les fonctionnalités collaboratives sont souvent présentées comme mineures. Même si le « partage » de la visualisation à la manière des « réseaux sociaux » est assez courant, l'édition collaborative est souvent complexe à implémenter car les besoins sont nombreux. Parmi eux, on trouve tout d'abord le besoin d'avoir un système distribué conservant la cohérence des modifications de chacun des utilisateurs. Or le nombre d'utilisateurs ne doit pas affecter l'expérience utilisateur ; le système doit supporter le passage à l'échelle. Un nombre d'utilisateurs élevé implique la mise en place d'un système de distribution de données adapté. Ce système est en plus contraint par la dynamicité des arrivées et départs des collaborateurs (*churn*) au cours d'une session. La dynamicité, qui ne permet pas d'avoir de pouvoir compter sur un nombre fixe de ressources, doit s'accorder avec les besoins variables en termes de ressources. Chaque client est porteur de ressources rarement complètement exploitées lors d'épisodes collaboratifs.

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

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

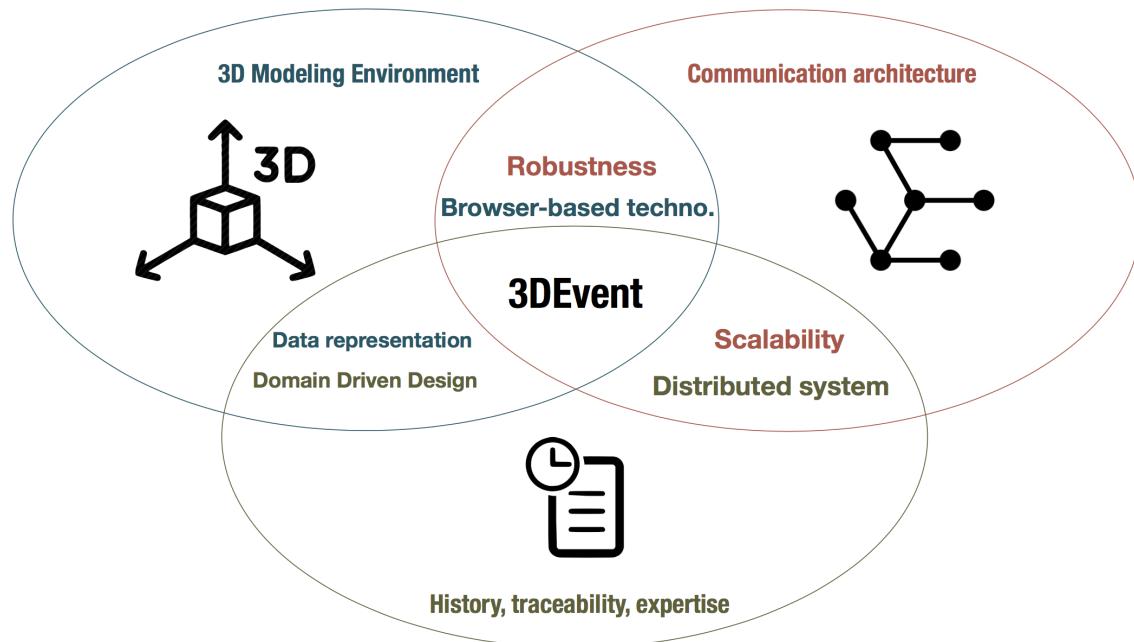


FIGURE 1.1 – Place de la contribution dans les différents champs de recherche

contraintes réseau, métier, et 3D dans un navigateur. Dans le but d'atteindre ces objectifs, voici les cinq Questions de Recherche posées :

- QR 1** Quelle architecture réseau est la plus adaptée pour une gestion efficace, robuste et temps réel des données 3D dans un environnement web ?
- QR 2** Quelle architecture logicielle confère une traçabilité des données conforme aux règles métier liées à la manipulation d'objet 3D ?
- QR 3** Quels sont les mécanismes assurant à l'utilisateur d'être autonome tout en ayant la possibilité de collaborer ?
- QR 4** Comment faciliter l'implémentation d'un tel système en garantissant le respect des règles métier liés à la manipulation d'objet 3D ?
- QR 5** Quelles sont les métriques (réseau, collaboration) permettant d'évaluer un tel système de manière quantitative ? Qualitative ?

1.3 Contributions

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

grâce à l'utilisation de solutions préexistantes pour le partage et la diffusion de contenus **3D** dans le cadre d'applications collaboratives sur le web.

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

1.3.1 Contributions théoriques

- Proposition d'un système de gestion et de visualisation de contenu **3D** avec un historique non linéaire orienté évènements.
- Définition d'une architecture hybride (client serveur et **P2P**) adapté à la distribution de contenus **3D** dans le cadre d'une collaboration sur web en temps réel.
- Introduction des concepts d'évènements métiers liées à la **3D** comme moyen d'interagir dans une scène **3D** multi-échelle

1.3.2 Contributions pratiques

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

1.4 Organisation du manuscrit

La **3D** et les environnements virtuels collaboratifs **3D** sont de plus en plus présents dans l'industrie de la **CAO**, ce qui rend nécessaires de nouveaux modèles, outils et méthodes facilitant la mobilité et l'autonomie des utilisateurs. Le contenu multimédia **3D** est également devenu omniprésent dans notre quotidien. De ce fait, développer de nouveaux outils pour créer manipuler et partager du contenu **3D** est crucial. Le chapitre présente une architecture de communication hybride permettant de faciliter la transmission des objets **3D** en temps réel de manière transparente pour

l'utilisateur. La création d'**EVC 3D** sur le web nécessite de nouvelles architectures tirant partie des nouveaux standards du [World Wide Web Consortium \(W3C\)](#). Puis le chapitre présente deux contributions reposant sur l'architecture cliente de 3DEvent pour manipuler des objets **3D** avec une grande traçabilité et de manière autonome. Finalement, le chapitre aborde le problème spécifique de la transmission du contenu **3D** lors d'une session collaborative en temps réel. En considérant et développant un modèle orienté évènements, 3DEvent dérive des stratégies pour délivrer et obtenir une scène 3D en continu (*streaming* en anglais) en s'appuyant sur les différentes configurations matérielles, utilisateurs et réseaux.

Chapitre 2

État de l'art

Contents

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 et au delà	20
2.1.3	Web 3D : deux approches	23
2.2	Communication en temps-réel	25
2.2.1	Fiable versus Non Fiable	27
2.2.2	Ordonné versus Désordonné	28
2.2.3	Les principaux protocoles de transport	29
2.2.4	Web et P2P : WebRTC	31
2.2.5	Quel protocole dans quel EVC3D ?	35
2.3	Les systèmes distribués orientés évènements pour la collaboration	36
2.3.1	Introduction	36
2.3.2	Systèmes Publish-Subscribe	37
2.3.3	Domain Driven Design	40
2.3.4	Command Query Responsability Segregation	43
2.3.5	Event Sourcing	44
2.4	Conclusion	49

Tableau 2.1 – Solutions conventionnelles pour les problèmes collaboratifs [103].

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

2.1 Modélisation 3D collaborative sur le web

2.1.1 Collaboration et concurrence en 3D

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

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

La plupart du temps, ce sont des mécanismes autoritaires comme le mécanisme de verrouillage (*lock mechanism*), des feux de signalement (*traffic light mechanism*) ou 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 plus restrictifs vis-à-vis de la liberté de création des utilisateurs du système. Par exemple, Lets3D [37] est un éditeur collaboratif 3D sur le web dont la gestion de la concurrence repose sur la demande de permission par objet sélectionné (ce qui verrouille l'objet pour les autres utilisateurs lorsqu'elle est accordée).

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 [30, 35]. Alors que Duplex [72], un **EVC 3D**, est présenté au même moment, ce dernier s'intéresse plus spécifiquement aux problématiques de passage à l'échelle, en essayant de réduire la taille du contexte partagé pour limiter les problèmes liés à la concurrence. Les modèles liés à l'annulation (*undo/redo*) dans les éditeurs collaboratifs distribués sont nombreux. Pour les documents textuels, les travaux les plus aboutis dans un environnement distribué collaboratif [77, 85, ?]. Les algorithmes souvent utilisés dans ce cadre sont les algorithmes de transformées opérationnelles – **OT** – [30] ou les algorithmes de réPLICATION de données commutatives – **Commutative Replicated Data Types (CRDTs)** – [79] s'orientant de plus en plus vers des solutions décentralisées pour un usage à grande échelle [96].

L'existence de protocoles agnostiques¹ comme Dat [70] permettent également une synchronisation fiable pour des contenus dynamiques. Dat est un protocole de synchronisation de jeu de données qui n'assume pas que le jeu de données soit statique ou que le jeu de données soit entièrement téléchargé. C'est un protocole agnostique dont la conception repose sur la garantie des principes suivants : intégrité du contenu grâce à l'utilisation de valeurs de hachage (*hash*) signées, mises à jour régulièrement de manière décentralisée (*decentralized mirroring*) pour découvrir automatiquement les pairs et effectuer l'échange de données en essaim, chiffrement des données de bout

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

en bout, versionnage incrémental pour la synchronisation des données. Ce protocole est, par exemple, utilisé par hashbase.io, un service qui se propose d'être un pair toujours présent pour l'hébergement de sites web en **P2P**, promettant ainsi une disponibilité pérenne du contenu.

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

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

La collaboration en temps-réel est de plus en plus présente sur le web pour différents types de documents, notamment 3D. La revue sur la visualisation distribuée, effectuée par Grimstead et al. en 2005 [36], indique que la plupart des systèmes sont conçus pour moins de cent utilisateurs simultanés et reposent sur un ou plusieurs

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

Les systèmes de **Réalité Virtuelle (RV)** collaboratifs font exception à ce schéma où l'utilisation des réseaux **P2P** est plus répandue pour supporter parfois plus d'un millier d'utilisateurs simultanés. Dans chacun des cas, chaque système a besoin d'un client adapté pour opérer de manière isolée, sans inter-opérer avec les autres systèmes. C'est dans ce contexte qu'en 2011, Mouton et al. [67] présentent une analyse approfondie de l'état 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).

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 aujourd'hui (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). 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 (*plugin less*), beaucoup sont développées pour faire de la modélisation 3D et utilisent l'architecture client-serveur et le protocole **WebSocket** pour effectuer la synchronisation entre les différents clients collaborateurs. Quelques raisons peuvent expliquer cette situation. Historiquement, les développeurs sont plus familiers avec les architectures client-serveur et le standard **WebSocket**. Technique, la gestion de données est facilité dans un architecture centralisée notamment concernant la synchronisation et la gestion de la concurrence dans un **Environnement Virtuel Collaboratif (EVC)**. 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 pour la modélisation **CAO** est prépondérant pour les solutions commerciales comme OnShape, Clara.io et TinkerCAD. Les fonctionnalités minimums de ces plateformes consistent en la mise

en place d'un environnement 3D visualiser un objet 3D accompagnées d'un système d'annulation / refaire et d'une plateforme de diffusion des créations. OnShape est un service de modélisation 3D très riche dont l'objectif est de proposer une qualité équivalente des fonctionnalités de modeleurs autonomes (*standalone*) professionnels, le tout de manière collaborative. Leur approche de la gestion de version développe le concept de micro version [5]. Cette approche est employée dans les travaux de Lu et al. [62] 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 [42] est plus généraliste. Il s'oriente davantage vers des fonctionnalités avancées de rendu 3D sur le *cloud* en utilisant du lancer de rayons. Les fonctionnalités d'historisation et de collaboration proposées dans Clara.io restent très basiques (seulement annuler / refaire). TinkerCAD est aussi une plateforme de publication d'objets 3D pour faciliter l'accès à la conception d'objets 3D. TinkerCAD est une application destinée au grand public pour la conception et l'impression 3D. De ce fait, les fonctionnalités sont limitées à cause du métier, l'impression 3D et de la cible, 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 surface implicite (BlobTree) [34]. 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 au types de modélisation 3D.

Tableau 2.2 – Encodage des données transmises

Référence	Modélisation 3D	Encodage des modifications
[34]	Blob (CSG)	Fonction implicite
Clara.io [42]	Polygonale	Commande interface
[68]	Polygonale	Fonction paramétrique
OnShape [5]	Paramétrique	Microversion
cSculpt [11]	Polygonale	Fréquence spatiale

2.1.3 Web 3D : deux approches

La plateforme web possède certains avantages par rapport à des clients lourds. En effet, le développement de l'infonuagique a permis le développement de meilleures infrastructures de services. Du fait de ce développement, la création de plateformes collaboratives sur le web pour faire de la conception 3D est devenue, non seulement faisable en termes de ressources, mais également en termes de technologie. Il existe deux types d'approches pour créer du contenu web 2D ou 3D. L'approche déclarative et l'approche impérative. La Figure 2.1 montre le pendant 3D pour chaque approche 2D. En 2D, les spécifications déclaratives ([Scalable Vector Graphics \(SVG\)](#)) et impératives ([canvas](#)) sont issues du HTML5, alors qu'en 3D, les spécifications sont encore en évolution.

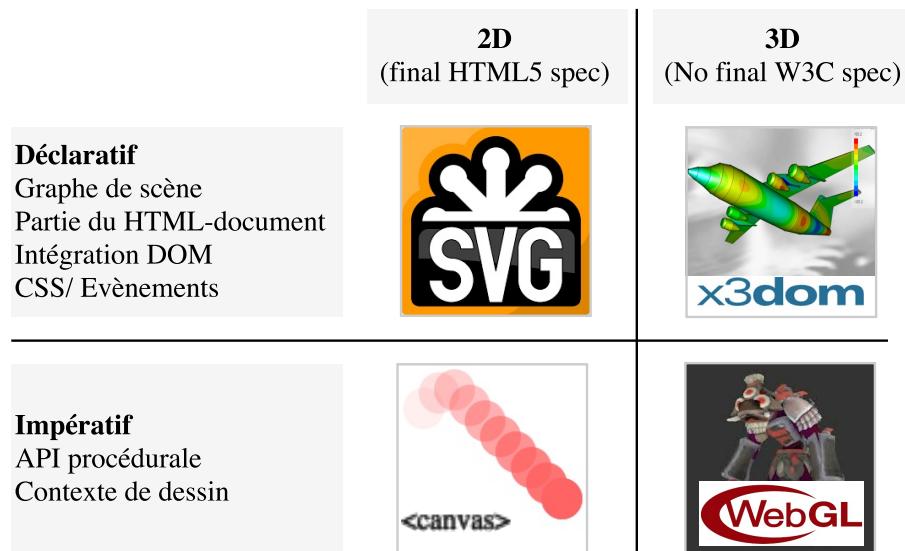


FIGURE 2.1 – Déclaratif vs Impératif en 2D et en 3D sur le web

L'approche déclarative (Dec3D) s'intègre au [Document Object Model \(DOM\)](#) et se focalise sur l'utilisation les technologies du web existantes comme CSS3, HTML5 et l'Ajax. X3D est ainsi un format de fichier respectant le standard ISO [95] 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 [7] et XML3D [81] : 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 en permettre l'intégration du format dans le [DOM](#). De plus, X3DOM intègre le support d'éléments [HyperText](#)

Markup Langage ([HTML](#)), les évènements [DOM](#) et de profils [CSS](#) en supplément [89]. 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 [45] pour des données spatiales [83] ou du rendu volumique [6], par exemple. 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 [32] ou les évènements du [DOM](#) [61] comme protocole de synchronisation de scènes. L'ajout de nombreux *listeners* sur un document peut affecter les performances de parcours de l'arbre du [DOM](#) notamment si une scène est très peuplée.



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

Concernant l'approche impérative, la spécification WebGL 1.0 [48] 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 [46] en utilisant l'élément *canvas* d'[HTML5](#). Cette spécification est supportée par la plupart des navigateurs traditionnels (Chrome depuis v49, Firefox depuis v52, Safari depuis v9.1, IE 11², Edge 14² et Opera depuis v41) ainsi que les navigateurs mobiles (iOS Safari depuis v9.3, Android Browser depuis v53 et Chrome for Android depuis v53) à l'exception

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

d'Opera Mini³. La spécification WebGL 2.0 [49] basée sur OpenGL ES 3.0 [47], déjà publiée en tant que brouillon, est en phase expérimentale⁴. 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 web⁵.

2.2 Communication en temps-réel

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

L'exigence vis-à-vis des temps de réponse est de plus en plus grande et ce pour deux raisons principales : les limitations humaines (mémoire et attention limitées à court terme) et les aspirations de l'être humain (besoin d'être en contrôle sur les machines). S'accroissant au gré de la technologie et des attentes des utilisateurs (rétro-compatibilité, temps de chargement), cette exigence varie selon le domaine : elle est prépondérante sur le web en général. Par exemple, dans un domaine connexe comme le e-commerce, une étude réalisée en 2009 explique qu'une grande partie des internautes abandonneraient leurs achats en ligne si les pages mettaient 2 secondes ou plus à charger⁶. De nos jours, ce délai a été réduit au quart de seconde pour les grandes entreprises du web. Jakob Nielsen [69] indique quatres valeurs limites concernant les temps de réponses :

- 0,1 seconde donne une sensation de réponse instantanée, comme si le résultat avait été produit par l'utilisateur et non l'ordinateur. Ce niveau de temps de réponse soutient la sensation de manipulation directe.⁷

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

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

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

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

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

- *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* conservent 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 d'autres choses, 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 s'accorder de délais relativement long (1-10s) compte tenu du fait que l'utilisateur connaît en partie ces contraintes liées à la taille des objets 3D. Concernant les mises à jour, l'édition collaborative requiert un temps de réponse raisonnablement court qui est contraint par l'exactitude des calculs et le temps dans lequel le résultat est produit. Pour les mises à jour internes – produites par l'utilisateur, actions sur l'interface – on s'accordera sur un délai inférieur à 0,1 seconde dans cette thèse. 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 dans un **Environnement Virtuel (EV)**, un jeu multi-joueurs ou un jeu sérieux (*serious game*), les participants interagissent en temps-réel ou quasi-réel, alors qu'ils sont situés à différents endroits géographiques. Le développement d'un **EVC 3D** nécessite donc des connaissances issues de plusieurs domaines comme la conception et l'implémentation de protocoles réseaux et de réseaux distribués.

Un protocole de communication en temps-réel existe, il s'appelle **Real-Time Transport Protocol (RTP)** et a été développé dans le but de faciliter la communication en temps-réel sur un réseau. Coexistant avec **RTP**, le protocole **Real-time Transfert Control Protocol (RTCP)** est utilisé pour transmettre des informations de contrôle à propos des données en temps-réel. **RTCP** transporte des données contenant des informations de contrôle et des informations statistiques concernant les flux de

données et les connexions des destinataires qui permettent à l'expéditeur d'ajuster le flux en conséquence. Les protocoles **RTP** et **RTCP** sont principalement conçus pour la transmission de flux audio et vidéo, moins pour des données arbitraires comme des données 3D. Pour cette raison, leur utilisation est limitée dans le cadre des **EVC** pour la modélisation 3D.

La variété des domaines et applications auxquels s'adressent les **EVCs 3D** actuels, fait qu'il n'existe pas de protocole de communication parfait, adapté à tous les types d'environnements. Les besoins en communication, qui se formulent souvent en termes de fiabilité de la livraison des messages et de l'ordonnancement des messages, sont les variables d'ajustement en fonction des contraintes qu'imposent les limites de temps de réponse et de bande passante.

2.2.1 Fiable versus Non Fiable

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

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

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

2.2.2 Ordonné versus Désordonné

Les paquets envoyés par l'expéditeur ne prennent pas forcément tous le même chemin vers l'expéditeur. 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 communications non ordonnées sont plus simples à gérer car les données sont délivrées dès qu'elles arrivent. Cela signifie également que la numérotation des paquets n'est pas requise (paquets plus légers). L'application est alors en charge de gérer le fait que les paquets arrivent dans le désordre. La livraison de paquets ordonnés est souvent une fonctionnalité appréciée dans le cadre des [Système d'édition collaborative \(SEC\)](#), et particulièrement en 3D, afin de respecter l'intention de l'utilisateur car l'ordre des actions détermine le résultat de ces actions. Par exemple, l'exécution de deux matrices 3D consécutives n'est pas commutative.

2.2.3 Les principaux protocoles de transport

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

TCP

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

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

UDP

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

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

Même si **UDP** n'est pas fiable et pas 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 empreinte des fonctionnalités à **TCP** mais utilise une approche orientée message comme **UDP**. Comme **TCP**, la connexion doit être négociée entre les deux terminaux. **SCTP** discute également le nombre de flux utilisés : il autorise le multiplexage de plusieurs flux de données dans une seule connexion **SCTP**. En comparaison, **TCP** sépare chaque flux dans une connexion différente : le support de plusieurs flux est géré à plus haut niveau.

SCTP a été conçu comme un protocole fiable avec une option pour envoyer les messages de manière ordonnée ou non ordonnée. L'ordonnancement choisi est signifié par un *bit-flag* dans l'en-tête associée au message. Une extension du protocole permet de configurer la fiabilité d'un message expédié. La fiabilité partielle peut être établie

en fonction de différents paramètres (temps d'attente ou nombre de tentatives avant retransmission). Cependant, seul l'expéditeur est au courant du niveau de fiabilité des données. Lorsque le destinataire remarque des données manquantes, il envoie à l'expéditeur des acquittements sélectifs (SACKs). Un SACKs contient les séquences de messages reçus et manquants et c'est à l'expéditeur de vérifier la configuration de la fiabilité des messages. Pour les messages marqués non fiables, l'expéditeur génère un nouveau paquet pour notifier le destinataire qu'il peut les « oublier » et avancer sa séquence de nombre à la valeur données dans le nouveau paquet.

Le champ de vérification de somme dans l'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[41].

Tableau 2.3 – Aperçu des protocoles de transport

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

2.2.4 Web et P2P : WebRTC

Le développement de la communication en **P2P** sur le web a débuté en 2011 avec les premiers brouillons du standard **Web Real-Time Communication (WebRTC)** proposé par le **W3C** et l'**Internet Engineering Task Force (IETF)**[®]. **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 continu audio et vidéo) mais très peu utilisent la partie

dédiée au transfert de données *DataChannel*. Selon le site officiel de WebRTC⁸, plus d'un milliard d'utilisateurs ont utilisé la technologie *open source* WebRTC. Ericsson Labs a été parmi les premiers à implémenter une application WebRTC en 2011. La jeunesse du protocole (2011) fait que peu de travaux académiques l'utilisent pour créer des **EVC 3D** [27, 82] et optimiser le partage de modèles 3D [52]. Côté commercial, un grand nombre d'applications et de services basés sur ce protocole ont fait leur apparition (par ordre d'importance) : des systèmes d'échanges audio / vidéo (VoIP)⁹, des systèmes de partage de fichiers¹⁰ et autres applications (capture d'écran, réalité augmentée, jeux)¹¹. WebRTC se retrouve également dans le domaine de l'optimisation de l'utilisation des ressources pour la création de **Content Delivery Networks(CDNs)** [105], de grilles de calcul comme dans browserCloud.js [28] ou pour faire des analyses visuelles collaboratives dans un environnement hétérogène [59].

L'utilisation et la gestion de réseaux **P2P** sont des changements de paradigme qui peuvent représenter une barrière d'entrée auprès des développeurs web qui sont 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) sont une motivation supplémentaire en plus des implications sociales que cette architecture (partage, coopération, coût) apporte en faveur de l'intégration de WebRTC dans les applications web.

Les machines clientes et les systèmes 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).

8. <https://webrtc.org>

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

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

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

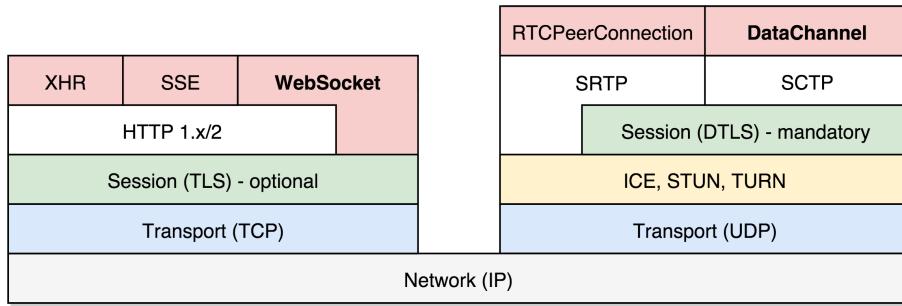


FIGURE 2.3 – Pile des protocoles IP : TCP vs UDP

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 de la livraison des paquets et sécurise nativement les données avec le protocole [Datagram Transport Layer Security \(DTLS\)](#). Le contrôle de la congestion des flux et l’ordonnancement éventuel est effectué par les protocoles [SCTP](#) et [Receiver-side Real-Time Congestion Control \(RRTCC\)](#).

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

Mécanisme de signalisation et problématiques réseaux

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

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

Un service de signalisation est nécessaire à la mise en relation des différents pairs au cours d’une session collaborative. Pour établir un canal entre deux navigateurs (initialiser une session WebRTC), il faut que chaque client se « signale » l’un à l’autre.

Techniquement, la connexion est établie en passant par un tiers ou directement entre deux pairs une fois qu'ils ont reçu leurs identifiants de connexion. Une fois que la connexion est établie entre deux pairs, RTCDataChannel est capable, en théorie, d'être le canal de signalisation. Cette solution peut aider à réduire la latence (car les messages sont envoyé directement entre les pairs) mais n'est pas très fiable. L'utilisation d'un tiers (serveur) est, en pratique, quasi systématiquement privilégiée.

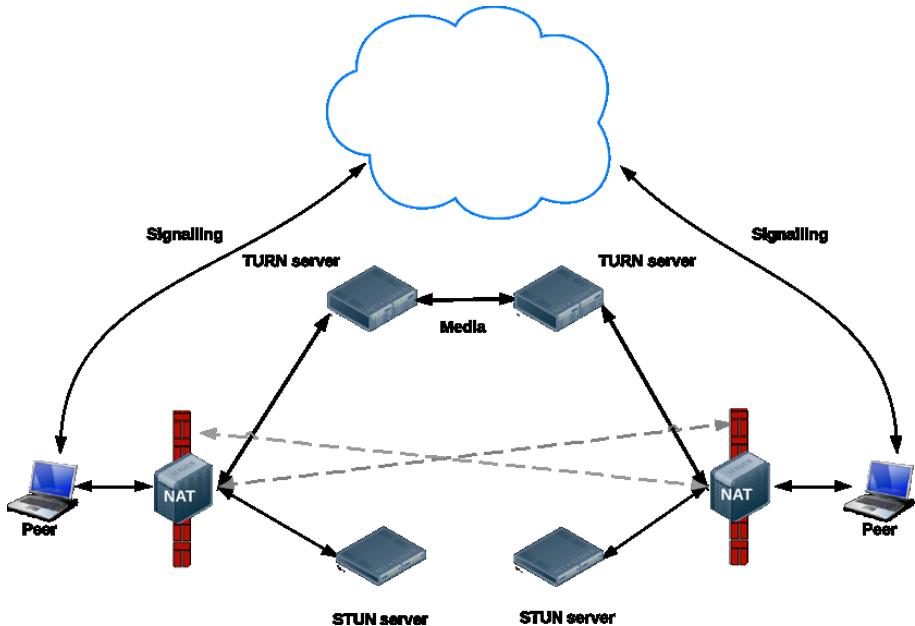


FIGURE 2.4 – STUN, TURN et signalisation

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, uniques et routables. Pour permettre à un client situé derrière un routeur **NAT** de connaître son adresse IP publique et le type de serveur **NAT** l'utilisation d'un serveur **Simple Traversal of UDP through NATs (STUN)** est requise. L'utilisation d'une **NAT** dynamique impose des restrictions sur l'adresse de destination. Dans ce cas, le **STUN** est inutile, on utilise un serveur **Traversal Using Relays around NAT (TURN)** qui est considéré comme une extension du **STUN** et fait office de proxy. Une fois le client parvenu à récupérer une adresse IP, il peut participer à l'initialisation d'une connexion

WebRTC (signalisation). L'utilisation de protocoles tels que [Interactive Connectivity Establishment \(ICE\)](#) avec son [ICE Framework](#) permet de faciliter la mise en relation des clients derrière un [NAT](#). [ICE Framework](#) 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](#)).

2.2.5 Quel protocole dans quel EVC3D ?

Un des aspects prépondérant dans les [EVC 3D](#) concerne la communication et la transmission des données lors de la collaboration. Que ce soit dans un [EV](#), un jeu multi-joueur ou un jeu sérieux (*serious game*), les participants interagissent en temps-réel ou quasi-réel alors qu'ils sont situés à différents endroits géographiques. Le développement d'un [EVC 3D](#) nécessite donc des connaissances issues de plusieurs domaines comme la conception et l'implémentation de protocoles réseaux et de réseaux distribués.

Comme l'indique [78], le protocole le plus souvent rencontré dans les [EVC 3D](#) est l'[UDP](#). Cette prépondérance s'explique d'après deux situations. Premièrement, dans des réseaux où l'accès est fiable car ces environnements n'ont pas (ou peu) de perte de paquet (ex : [LAN](#)). Deuxièmement, dans un contexte d'application où la perte de paquet n'est pas critique (ex : jeu vidéo, diffusion en continu de son, vidéo, 3D...). [UDP](#) peut également s'utiliser dans tous types de réseaux – [LAN](#) ou [WAN](#) (comme internet) –, car il est basé sur des datagrammes. Le fait qu'il n'effectue pas de vérification de délivrance lui donne un avantage quant à la taille des données envoyées dans les [EVCs](#). La responsabilité de vérifier quels sont les paquets qui n'ont pas été délivrés et leur ordre d'arrivée incombe à l'application.

[TCP](#) est le protocole qui vient en seconde position. Les [EVC 3D](#) qui sont implémentés sur des réseaux haut-débit ne sont pas affectés par les étapes de vérifications qui nécessitent une connexion [TCP](#). D'après [88], 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 notamment dans l'assistance par les pairs pour le rendu est une méthode qui permet aux pairs hébergés sur des appareils avec des capacités limitées (bande passante, processeur, processeur graphique) de demander une partie du contenu de l'environnement aux autres pairs. Des systèmes récents comme celui de Martinez et al. [64], utilisent la localisation d'un utilisateur pour transmettre uniquement les mises à jours à son plus proche voisin. Ce système d'assistance par les pairs peut également être utilisé pour améliorer le rendu dans les environnements virtuels en ajustant la qualité du rendu en fonction du coût de calcul [106]. Koskela et al. [52] 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 [29] et du BIM [18] 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 évènements pour la collaboration

2.3.1 Introduction

La plupart des systèmes basés évènements sont très populaires lorsqu'il s'agit de collaboration[39]. Les évènements peuvent être utilisé dans Computer-Supported Cooperative Work (CSCW). On retrouve par exemple la spécification d'évènements composites pour le support de la collaboration dans la conception logicielle [104]

ou encore la définition de patrons de conception dédiés à la collaboration dans les architectures basées évènement [92]. Dans ce dernier domaine Papageorgiou et al. [73] proposent un assistant pour la création de ces patrons pour la collaboration en s'appuyant sur un système de recommandation basé sur le contexte qui utilisent une représentation sémantique ([Web Ontology Language \(OWL\)](#)).

CoDesign est un exemple de [framework](#) permettant de faire de la conception logicielle de manière collaborative [4]. L'utilisation d'une architecture basée évènement 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 concurrentes 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 distribuée inhérents aux architectures microservices en utilisant [ES](#) et [CQRS](#). Une de leurs application exemple est un tableau Kanban collaboratif temps-réel construit sur leur plateforme ¹².

2.3.2 Systèmes Publish-Subscribe

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

Les modèles utilisant le paradigme [PubSub](#) supportent naturellement une communication *many-to-many* entre les *publishers* et les *subscribers*. De cette manière, on permet un meilleur passage à l'échelle d'une application avec une topologie du réseau plus flexible.

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

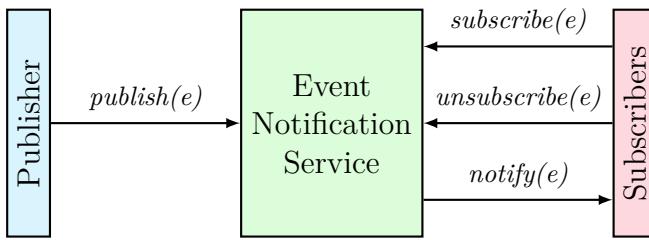


FIGURE 2.5 – Architecture Publish-Subscribe

Pour répondre à des besoins de mise à l'échelle, d'interopérabilité, de fiabilité, d'expressivité et d'utilisabilité, il s'appuie sur un modèle **PubSub** basé types et attributs. En spécifiant d'abord le type et ensuite en filtrant sur les attributs des évènements, ce modèle rend le routage des données intuitif pour le développement d'application distribuées à grande échelle comme le commerce en ligne (*e-commerce*). Scribe [15] et Hermes [75] sont des exemples de systèmes **PubSub** 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'évènement dans la couche réseau et construisent et maintiennent la distribution à partir de ce nœud *rendezvous*. TERA [2] propose une dissémination sur deux couches orientée sujet. Le but est d'effectuer une distribution uniforme en fonction des intérêts des nœuds. La dissémination se déroule en deux phases. Pendant la première phase, un algorithme de marche aléatoire (succession de pas aléatoires dans le graphe) est utilisé sur la couche basse qui connecte les nœuds dans le but de localiser le nœud qui a souscrit au sujet. La seconde phase commence lorsqu'un *subscriber* est trouvé pour le connecter au *cluster* correspondant au sujet sur la couche haute. C'est là que l'évènement est disséminé.

En juin 2017, Leonardo Querino présente une rétrospective des systèmes **PubSub** 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 [9] 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. » [9]

Après cette publication les premiers systèmes **PubSub** commencent à se développer. Le concept de routeur d'information est introduite par Oki et al. [71] : 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 capable d'introspection (service, opérations, attributs) pour adapter leur comportement au changement, les types peuvent être définis dynamiquement et la communication est anonyme. Puis l'apparition de solutions plus avancées sont apparues avec le projet GRYPHON d'IBM [3], SIENA [13], REBECA [74] et également REDS, JEDI, PADRES présentés plus en détails dans [90]. Ces différentes solutions ont été conçues comme des systèmes de gestion avec des gestionnaires d'évènements dédiées. Le passage à l'échelle est principalement effectué par un routage efficace des évènements (comme le filtrage d'évènements). Seuls quelques travaux ont utilisé le concept de *clustering* par intérêt comme moyen afin de réduire le nombre de notifications d'évènement et éviter la surcharge. A partir des années 2000, les systèmes **P2P** ont commencer à émerger avec l'apparition de Gnutella et consorts, les **DHTs** (Chord, Pastry, Kademlia) et Réseaux non-structurés (Cyclon, Scamp, ADH). Ces systèmes ayant pour objectifs de gérer des réseaux massifs, dynamiques (*churn*), résistants aux erreurs, et offrant des primitives de communications basiques. Les réseaux **P2P** promettent alors des propriétés intéressantes pour les infrastructures logicielles comme les systèmes **PubSub** concernant le passage à l'échelle. Cependant, les systèmes **PubSub** d'alors reposent sur une connaissance totale du réseau pour être efficace, du fait de leur taille limitée. La connaissance globale de l'information ne peut être collectée dans un système à grande échelle, c'est pourquoi la décentralisation de l'information est nécessaire. La connaissance peut vite devenir viciée dans un système dynamique décentralisé ce qui implique de trouver une méthode efficace pour gérer la dynamicité.

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

Outils de surveillance, contrôle de performances et validation ergonomique

Dans un système distribué comme [PubSub](#), les outils de comparaisons sont souvent très hétérogènes et ne permettent pas forcément d'évaluer sur une même base la charge en charge. Carzaniga and Wolf [14] proposent quelques lignes directrices pour concevoir une suite de *benchmark* dans un système [PubSub](#), 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 de liés à l'[Interface Utilisateur \(IU\)](#) :

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

Pour évaluer la charge de travail d'un système distribué basé événements, Kounev et al. [54] ont utilisé des techniques d'analyse opérationnelle pour caractériser le trafic du système et dériver une approximation de la moyenne des délais de livraison d'événements.

Beaucoup de recherches ont été menées dans le but d'améliorer les performances des systèmes collaboratifs utilisant une architecture distribuée. Bien que ces approches 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 la foule (*crowd computing*) [59]. De ce fait, le contenu 3D est distribué par les pairs et non plus par le serveur. En combinant les ressources réseau du serveur et de plusieurs clients, on peut réaliser une distribution des données (événements) en améliorant les performances du système pendant les sessions de travail collaboratif.

2.3.3 Domain Driven Design

[Domain Driven Design \(DDD\)](#) ou Conception Pilotée par le Domaine est une approche de développement logiciel qui a pour objectif de définir une vision et

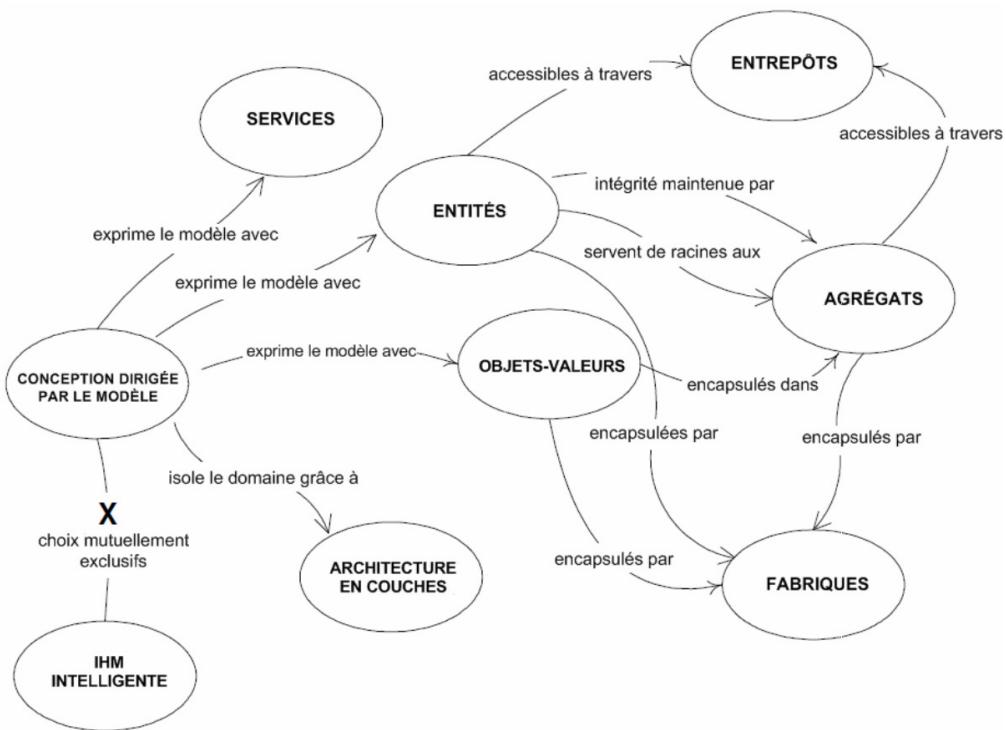


FIGURE 2.6 – Illustration du patron DDD et de ses artefacts (issue de [1] sous licence Creative Commons)

un langage partagé (*ubiquitous language*) pour les personnes impliquées dans la construction d'une application [31]. Le but est de mettre l'accent principal d'un projet sur le domaine et la logique du domaine en basant des conceptions complexes sur un modèle du domaine afin d'initier une collaboration créative entre les experts techniques et les experts du domaine pour raffiner itérativement un modèle conceptuel qui relève de problèmes spécifiques au domaine. Les différents concepts du DDD sont listés en suivant :

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

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

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

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

Le DDD permet de connecter le modèle et son implémentation en offrant plusieurs avantages. La plasticité du système est mise en avant par l'expression de règles

et de comportements qui facilitent les changements fréquents. L'accent mis sur l'identification des interactions dans le système encourage la mise en œuvre d'une interface orientée tâches (*task-based UI*). La testabilité fonctionnelle est intégrée via les règles métier explicitées et concentrées dans une couche spécifique de l'application. Cela les rend plus facilement identifiable et testable automatiquement. La robustesse est améliorée face aux changements dans le système d'information.

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

- **L'entité** (*entity*) : un objet qui n'est pas défini par ces 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 maillage qui sont réutilisés alors une maillage est considérée 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 objet-valeur qui ne seront jamais modifié car les utilisateurs ne sont intéressés que par les informations qu'elles portent.
- **L'agrégat** (*aggregate*) : une collection d'objets qui sont liés ensemble par une entité commune (*root entity*), connue sous le nom d'agrégat souche (*aggregate root*). Ce dernier garantit la cohérence des modifications faites au sein de l'agrégat en interdisant aux objets externes de faire référence à ses membres. Par exemple, dans une modélisation 3D collaborative un utilisateur ne peut pas modifier le nom d'un autre utilisateur. Un maillage n'a également pas d'intérêt à connaître les informations des utilisateurs, c'est la scène qui gère l'interface entre les maillages et les utilisateurs.
- **L'évènement du domaine** (*domain event*) : un évènement 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 [31] et [93]¹³.

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

Les disciplines liées à la 3D dans un contexte industrielles ont besoin de pouvoir communiquer sur un langage commun pour permettre à chacun des intervenants de s'exprimer dans la création du produit. Par exemple, la **CAO** est très utile dans un contexte d'ingénierie par 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 le flux d'écriture et de lecture dans le logiciel.

Introduit par Greg Young en 2009 [101], **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 la (*scalability*) en faisant apparaître de nouvelles forces : la gestion de la concurrence dans la collaboration sur des règles métier propre à la modélisation 3D dans des cadres d'application spécifiques. En effet, selon le type d'application, un ensemble de règles régit les droits concernant les types de modifications acceptables et qui est autorisé à le faire.

Le caractère vicié (*staleness*) d'une donnée dans un environnement collaboratif est récurrent. Une fois que la donnée a été montrée à un utilisateur, la même donnée peut être changée par un autre utilisateur, elle est altérée. **CQRS** pallie cela en répondant aux besoins suivants :

13. Les différents livres 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.

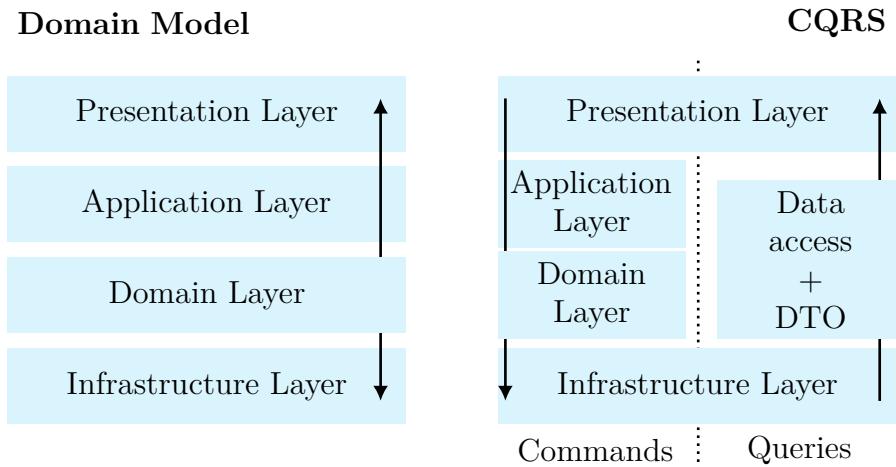


FIGURE 2.7 – Architecture en 4 couches du DDD (gauche) en miroir avec l'architecture CQRS (droite)

- 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 évènement. Les commandes sont immuables car leur usage nécessite un envoi direct au domaine pour être traitées. Quant aux évènements, ils sont immuables car ils représentent ce qui s'est produit dans le passé (qu'on ne peut donc pas changer).

2.3.5 Event Sourcing

Le Reactive Manifesto, apparu en 2014, est un document qui résume les propriétés clés liées aux systèmes distribués et encourage notamment le développement de systèmes « plus flexibles, à couplage faible et extensibles »[?] 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 évènements en mode ajout seulement (*append-only*) pour sauvegarder le résultat des commandes sur le domaine. Cela permet de conserver tous

les changements qui ont mené à un état plutôt qu'uniquement l'état. Ce paradigme permet de recréer n'importe quel état d'un agrégat à partir de la liste d'évènements qu'on lui a appliqué. Cette liste représente une base la vérité du système. **ES** permet de simplifier les tâches complexes dans des domaines complexes comme la 3D en ne requérant pas la synchronisation de modèle de données et du métier.

L'**ES** est souvent présent dans des architecture asynchrones qui ont l'avantage de pouvoir utiliser des queues de message, plusieurs bases de données et où la partie lecture est éventuellement consistante. La plupart de la littérature concernant ce patron se trouve en ligne, dans des billets de blog, des présentations, ou de la documentation logicielle. La littérature académique est relativement réduite, souvent rapproché des travaux sur l'évolution de graphes. Cette section fournit un aperçu des différentes définitions données de l'**ES** et du vocabulaire lié à ce patron de conception.

Martin Fowler 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 ». « série d'évènements capture tout ce qui est nécessaire à la reconstruction de l'état courant ». Il voit les évènements comme immuables et le journal d'évènement (*event log*) comme un stockage linéaire (*append only store*) des évènements (Figure 2.8a). Les évènements ne sont jamais supprimés, le seul moyen de l'"annuler" consiste à effectuer générer un évènement rétroactif. Une fois qu'un évènement rétroactif est ajouté, l'évènement rétroactif agit à l'inverse de l'évènement précédent pour compenser ses effets (Figure 2.8b). Dans ce billet, Fowler n'établit pas clairement la distinction entre les évènements et les commandes qui déclenchent ces évènements. Ce problème est considéré dans plusieurs travaux fondamentaux [77, 85, 96, 97] approfondie par un revue de Chen et al. [20].

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

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

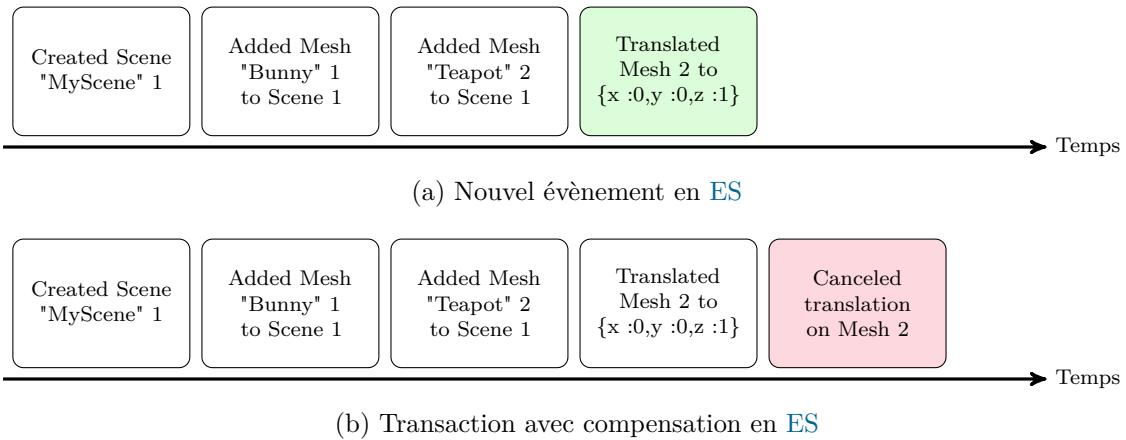


FIGURE 2.8 – Transaction en ES

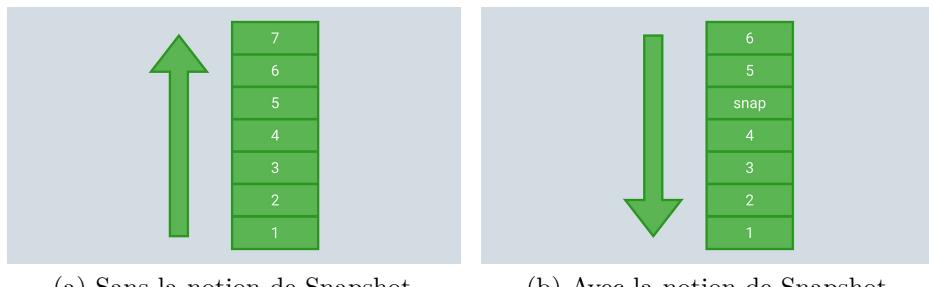


FIGURE 2.9 – Snapshot en ES

Les avantages et les inconvénients de l'approche **ES** dépendent du cadre dans lequel elle est utilisée. En reprenant ceux cités par Klamer [50], ils peuvent être envisagés sous l'angle de la modélisation 3D collaborative.

Avantages

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

«dans le passé» et récupérer les données à partir de ce moment. Par comparaison, les avantages de l'**ES** sur l'Active Record sont :

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

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

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

Inconvénients

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

Un **Snapshot** est une capture de l'état de l'agrégat à un moment qui correspond à l'empilement d'évènements ayant mené à cet état. En créant un **Snapshot**, on évite de reconstruire un état à partir du début de la vie de l'agrégat (Figure 2.9a) ; on empile les nouveaux évènements à partir de ce snapshot (Figure 2.9b). L'utilisation du **Snapshot** est intéressante à partir du moment où la pile d'évènements 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'évènements).

Le parcours des évènements 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 évènements qui sont arrivés entre ce **Snapshot** et l'état courant. Dans le cas où l'on souhaite accéder à des données historiques plus anciennes que le dernier **Snapshot**, le second parcours fonctionne aussi. Cependant, elle doit être évitée pour ne pas créer de dépendance entre les **Snapshot**. Sans les snapshots le modèle peut encore varier « librement » tant que l'on sait comment lui appliquer un évènement 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 évènement représente quelque chose qui est arrivé dans le domaine. Un évènement appliqué sur un état, donne un nouvel état. La condition pour appliquer un pur déterminisme en **ES** est la fonction suivante : **État** → **Évènement** → **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'évènement tous les informations lui permettant de faire la transition (i.e. sans effet de bord). L'évènement 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 évènements (qui ne sont pas forcément appliqués à l'état courant de l'application par la suite) : **État** → **Commande** → **Liste d'évènements**.

La différence principale entre un évènement 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 évènements, tandis que l'**ES** se contente d'appliquer un changement à l'état courant. De plus, en **ES**, l'évènement stocké a été produit par l'agrégat. La différenciation majeure des deux patrons de conception s'accentue lors l'interaction avec des systèmes externes. L'aspect fonctionnel de l'application de changement d'état de l'**ES** al'avantage de permettre de reconstruire l'état sans effets 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 de base de données comme EventStore ou le langage 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 évènements produits par les agrégats ne modifient l'état interne de l'application.

2.4 Conclusion

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

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

mode non fiable et non ordonné 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 évènements. Le couplage lâche est une propriété qui s'accorde bien avec les systèmes distribués P2P ou hybride. Elles 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 évènements dans un système collaboratif permettent d'intégrer les règles liées au métier dans une application de modélisation collaborative. Dans un premier temps, l'architecture PubSub, 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'évènement 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 une vision centrée métier au logiciel. 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 facilite la restitution des données sans risques d'effets de bord. Le côté écriture, permet de valider le contenu entrant selon les règles métiers du le domaine métier. L'ES est le patron qui introduit la notion d'évènement (« ce qui s'est passé »). L'état de l'application se reconstruit à partir de ces évènements issus de l'expertise métier – déduite grâce au DDD. Les évènements générés par CQRS sont stockés dans le journal des évènements qui les partage aux autres collaborateurs, ce qui peut provoquer des conflits qu'il faut pouvoir détecter pour éviter les incohérences dans l'EVC.

Chapitre 3

Contributions scientifiques

Contents

3.1	Introduction	52
3.2	Modèle évènementiel 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	Cohérence Éventuelle en CQRS	61
3.2.4	Potentielles applications et autres utilisations	63
3.2.5	Bilan	64
3.3	Architecture de communication hybride	64
3.3.1	Présentation générale	66
3.3.2	Extension : Event Store distribué	68
3.3.3	Persistance à long terme	69
3.3.4	Synchronisation client-serveur	69
3.3.5	Mécanisme de gestion de version	70
3.3.6	Gestion de la cohérence	71
3.3.7	Bilan	71
3.4	Conclusion du chapitre	71

3.1 Introduction

Les **EVC 3D** doivent considérer beaucoup de critères pour proposer un système réactif, robuste et permettant le passage à l'échelle en intégrant les contraintes liées à la gestion de données 3D. La dimension collaborative ajoute les problématiques de gestion de données concurrentes. En s'intéressant d'abord aux spécificités liées à l'assemblage d'objets 3D dans une scène partagée cela permet de mieux comprendre les fonctionnalités spécifiques à la 3D (transformations), à l'historisation et à la visualisation des données. De plus, les données doivent suivre un flux qui permet de vérifier la validité des données générées. 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 présente les contributions scientifiques de cette thèse. Tout d'abord, je présente le modèle orienté événements pour la visualisation et la manipulation d'objets 3D dans un environnement web. La contribution insiste sur l'intégration de la partie métier de la 3D par l'utilisation de patrons de conception (**DDD**, **CQRS**, **ES**). La réflexion concernant le découpage des opérations 3D liées à la manipulation d'objets 3D est expliquée dans un premier temps. Dans un second temps, pour procurer plus d'autonomie aux utilisateurs, je propose 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 ne permet pas l'échange entre les utilisateurs bien qu'elle soit pensée pour intégrer les aspects métiers de cette dernière.

La seconde contribution s'intéresse à l'architecture de communication dans un environnement web. L'architecture de communication doit permettre au système d'être résilient, léger et transparent. L'architecture hybride proposée repose sur une combinaison de l'architecture client-serveur et l'architecture **P2P**. Cela tient du fait que la centralisation de l'information est nécessaire dans un contexte industriel et que les propriétés du **P2P** permettent une haute disponibilité des ressources. Cette contribution est découpée en deux parties. La première concerne la présentation d'une preuve de concept sur la base du modèle présenté dans [27] 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 [26] et utilise la méthode de distribution des informations présenté dans [24] pour améliorer la résilience du système.

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

La méthodologie orientée évènements 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 évènements peuvent en effet être utilisés dans le cadre de **CSCW** et d'**EVC**, qui ont évolués du simple collecticiel à de plus importantes organisations virtuelles comme on peut le 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 évènements et donner des informations à propos du travail collaboratif. En utilisant les évènements suivis, les services d'analyse fournissent des statistiques à propos des collaborations passées et présentes.

Un système orienté évènements a l'avantage d'être découpé. 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 qu'elle apporte est aussi l'occasion de s'intéresser à la façon dont sont générées les données produites par les utilisateurs et la façon dont elles sont affichées.

L'**event processing (EP)** occupe, dans le champ de l'informatique, une place centrale dans beaucoup de systèmes comme l'énergie, la santé, l'environnement, les transports, la finance, les services et l'industrie. L'**EP** réunit des méthodes et des outils pour filtrer, transformer, et détecter des motifs dans des évènements, dans le but de réagir à des conditions qui changent, généralement liées à des contraintes de temps [17]. L'**EP** intègre plusieurs fonctionnalités pour :

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

— et surveiller (*monitor*) l'exécution de cette réponse.

Le panorama d'applications et de technologies proposé par [40] permet de définir les termes et les interprétations communes à différents domaines se basant sur le paradigme de l'[EP](#).

Constat

Par nature, une [EDA](#) est extrêmement peu couplée et hautement distribuée. Le créateur de l'évènement sait seulement que l'évènement se produit et n'a aucune idée du traitement que l'évènement va subir par la suite ou 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 évènement, elle n'en demeure pas moins complexe selon l'échelle d'évaluation. A 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 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 au [cloud](#). Le serveur a une puissance de calcul plus importante (et surtout ajustable).

Côté client, la puissance de calcul des machines sur lesquelles sont installés les navigateurs web évolue rapidement (notamment les appareils mobiles comme les *smartphones* et les tablettes). Les navigateurs suivent cette tendance en puisant dans ces ressources pour effectuer des traitements similaires à ceux que l'on trouve traditionnellement côté serveur et pour proposer des fonctionnalités avancées telles que le stockage de données sur le client (IndexedDB, storageAPI), pour l'affichage 3D (WebGL) et pour la communication en P2P ([WebRTC](#)). En déportant ainsi la charge que pourrait subir une architecture client-serveur côté client, les échanges réseaux

sont limités car ils sont très coûteux d'un point de vue énergétique pour les appareils mobiles [51]. De plus, l'utilisation de la bande passante est onéreuse et parfois limitée voire inexiste, il est donc nécessaire de tirer parti de tous les appareils participant à la collaboration au lieu de tout faire reposer sur le serveur. Chaque appareil participant à la collaboration doit être autonome et le plus indépendant possible en termes de ressources (données, réseaux, validation experte).

Contribution Pour répondre à cette problématique, je propose un modèle pour la transmission des données 3D et collaboratives qui permet de limiter le nombre de requêtes et la taille des données transmises sans perdre la traçabilité de celles-ci. L'idée est de profiter de la puissance du client pour créer une architecture assurant l'autonomie de l'utilisateur en cas de déconnexion volontaire (travail hors ligne) ou involontaire (coupure). Je garantis ainsi à l'utilisateur l'utilisation du système avec un historique performant où chaque connexion est l'occasion de mettre à jour le système.

3.2.1 Modèle général

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

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

Spécification des évènements dans le framework

Le framework étant orienté évènements, cette section de thèse introduit la représentation et le vocabulaire utilisé dans la description des évènements de 3DEvent. 3DEvent reprend la représentation proposée par Tominski [91] illustrée par la Figure 3.1, et adapte ses propriétés pour la modélisation 3D collaborative (visualisation et manipulation).

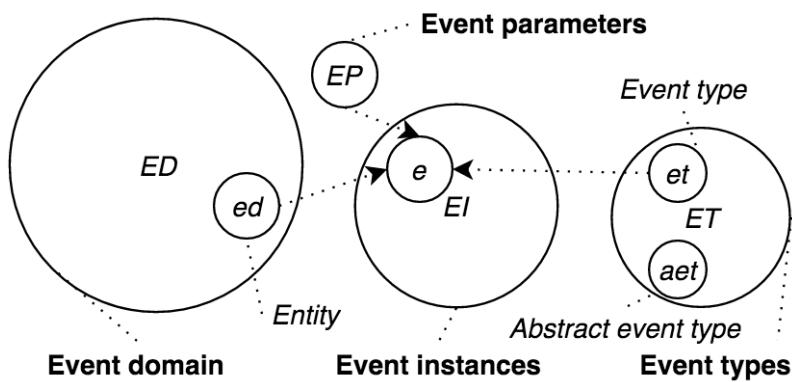


FIGURE 3.1 – Illustration des notions de évènements du domaine, des types d'évènements, et des instances d'évènements.

Pour décrire le cadre de référence dans lequel les évènements se produisent, la notion de domaines d'évènements est introduite. Un domaine d'évènements *ED* (*Event Domain*) contient les évènements qui respecte les types d'évènements qui peuvent être 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'évènements *ET* (*Event Type*) particuliers. Un *ET* est utilisé pour exprimer un intérêt concret envers les entités d'un *ED*. Les évènements abstraits *aet* (*abstract event type*) sont compatibles avec l'*ED*. Une instance d'évènement *e* – ou simplement évènements – est composée du triplet *ED*, *ET* et *EP*. L'ensemble des toutes les instances d'évènements possibles est exprimée par *EI*. *EP* (*Event Parameters*) correspond à l'ensemble des paramètres que peut avoir l'*EI*. Ainsi, tous les évènements sont liés aux intérêts d'un *ED*. La spécification des évènements nécessite de s'intéresser à tous les types d'évènements qui sont utilisés lors de la manipulation et la visualisation d'objets 3D dans un [EVC](#). La détection d'évènement est un processus qui permet de récupérer les instances d'évènements d'un *ET* en particulier. Cela peut être utilisé pour la visualisation de types particulier par exemple. Dans 3DEvent, la spécification des évènements est calquée sur les comportements observés lors des expérimentations des premiers travaux [27, 25].

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

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

Évènement	Nommage	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 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

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 nommé « Maillage » réagit à un certain nombre de types d'évènements comme *meshAdded*, *meshDropped* ... Les évènements typés sont produits par l'agrégat en prenant en compte les paramètres nécessaires à leur instanciation. Par exemple, la production d'un évènement *meshTranslated* prend en paramètre 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'évènements pour faciliter l'implantation de nouveau types d'évènements au plus proche des besoins des utilisateurs dans une application de modélisation 3D collaborative (appuyant la sémantique des manipulations).

3.2.2 Adaptation des patrons ES et CQRS

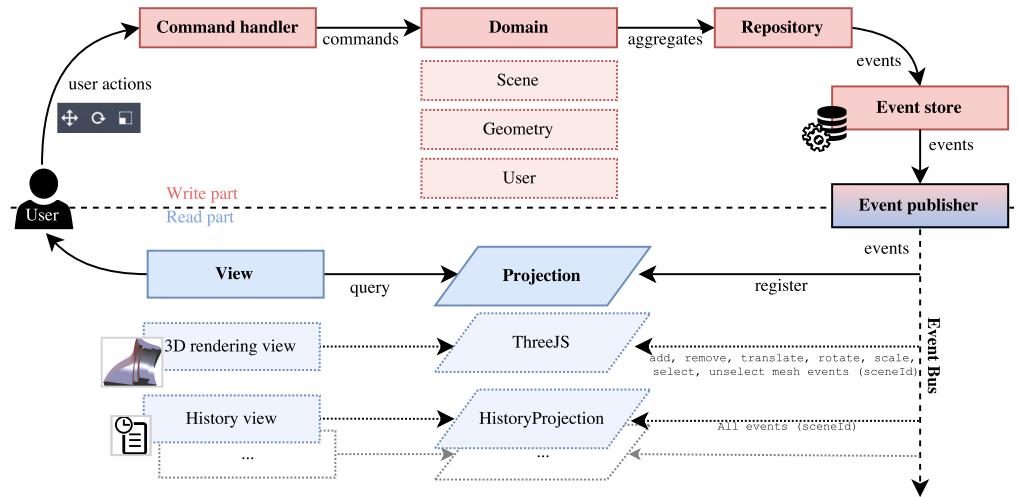


FIGURE 3.2 – Modèle de l’architecture client dans 3DEvent : la gestion du cycle de vie des données avec CQRS et ES dans un navigateur web, des actions utilisateurs à la visualisation en passant par la synchronisation réseau. Issu de [24].

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

Le patron Event Sourcing (ES) permet de capturer tous les changements d'état d'une application sous la forme d'une séquence d'évènements. Ces évènements sont conservés dans un journal d'évènements et peuvent être rejoués pour retrouver l'état de l'application. Les évènements représentent des faits immuables qui sont seulement

ajoutés au journal les un après les autres, ce qui permet des taux de transaction élevés et une réPLICATION efficace (cf Section 2.3.5). Dans 3DEvent, plusieurs composants d'**ES** sont étendus selon les applications :

- **Acteur** Un acteur consomme des évènements à partir d'un journal d'évènements et produit des évènements pour le même journal d'évènements. L'état interne dérivé à partir des évènements 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 évènements à partir du journal d'évènements. L'état interne dérivé à partir des évènements 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 évènements à partir du journal d'évènements pour mettre à jour la base de données. L'état interne dérivé à partir des évènements 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 évènements à partir d'un journal d'évènements et produit les évènements traités pour un autre journal d'évènements. Les processeurs peuvent être utilisés pour connecter les journaux d'évènement au traitement des évènements..

Le journal d'évènements Les évènements produits par un des composants présenté ci-dessus peuvent être consommés par d'autres de ces abstractions s'ils partagent un journal d'évènements local ou distribué. Un journal d'évènements 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'évènements local même s'il est partitionné sur plusieurs sites. Les journaux d'évènements locaux situés sur plusieurs sites peuvent être connectés par le biais d'un journal d'évènements dit « répliqué » (copié sur une autre réplique) qui a pour responsabilité de préserver l'ordre causal des évènements. 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'évènements local. Les composants peuvent soit collaborer sur un journal d'évènements local sur le même site, ou bien au travers d'un journal répliqué sur différents sites. Il est important de différencier le journal d'évènements 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'évènements, cependant et bien que parfois confondus, ils restent bien distincts conceptuellement. Le journal d'évènements commun est la base des échanges pour communiquer par le biais d'évènements de collaboration. Ce type d'architecture se retrouve dans différents cas d'utilisation :

- *Processus métier distribués.* Les acteurs de différents types utilisent des évènements 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'évènements (programmation réactive) en mettant à jour l'état de l'application et en produisant de nouveaux évènements. Cette forme de collaboration est appelée collaboration dirigée par les évènements.
- *RéPLICATION d'état d'Acteur.* Les acteurs de même type consomment les évènements 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 conflit en cas de mises à jour concurrentes et conflictuelles.
- *Agrégation d'évènement.* Les vues et les producteurs agrègent des évènements à partir d'autres composants pour générer des vues spécifiques à l'application. La collaboration évènementielle apporte de la fiabilité dans 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 les répliques sont à jour.

Les composants souscrivent à leur journal d'évènements en s'accrochant au **bus d'évènements**. Les évènements nouveaux sont poussés vers les souscripteurs, ce qui leur permet de mettre à jour l'état de l'application avec une latence minimale. Un évènement écrit à un endroit est publié de manière fiable aux souscripteurs sur ce site et aux souscripteurs des sites distants. Par conséquent, les composants qui échangent par le biais d'un journal d'évènements répliqué communiquent via un bus qui préserve l'ordre causal des évènements 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 d'écrire des évènements localement. La livraison des évènements sur les sites distants reprend automatiquement lorsque les partitions sont à jour.

Le journal d'évènements est répliqué localement et fournit un ordre total des évènements stockés et appartient à un site. Le site est une zone de disponibilité qui héberge un ou plusieurs journal d'évènements. Les évènements d'un journal d'évènements sont répliqués sur les sites distants de manière asynchrone. Afin de lier des journaux d'évènements (localisés sur différents sites) à un journal d'évènements répliqué, les journaux d'évènements 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'évènements répliqué est représenté par un journal d'évènements local sur chacun des sites participants.

Les points d'entrée permettent de gérer un ou plusieurs journaux d'évènements. 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 évènements stockés : l'ordre de stockage est le même sur tous les sites, ce qui veut dire que les consommateurs qui lisent les évènements du journal local vont toujours voir les effets avant leurs causes.

3.2.3 Cohérence Éventuelle en CQRS

La **cohérence éventuelle (CE)**, ou *eventual consistency* en anglais, 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 évènements durant les mises à jour reste identique lorsque les évènements sont rejoués par la suite car l'ordre issu de l'**ES** est déterminée par l'ordonnancement des évènements stockés localement. Au sein d'une réplique, tous les composants **CQRS** respectent cet ordonnancement. L'ordre de stockage des évènements répliqués sur des sites distincts est cohérent et ordonnés de manière causale [55] : les évènements liés causalement ont le même ordre sur tous les sites alors que les évènements concurrents peuvent avoir un ordre différent. Cette propriété est importante pour obtenir une cohérence causale forte dans une application qui respecte le théorème **Consistency**,

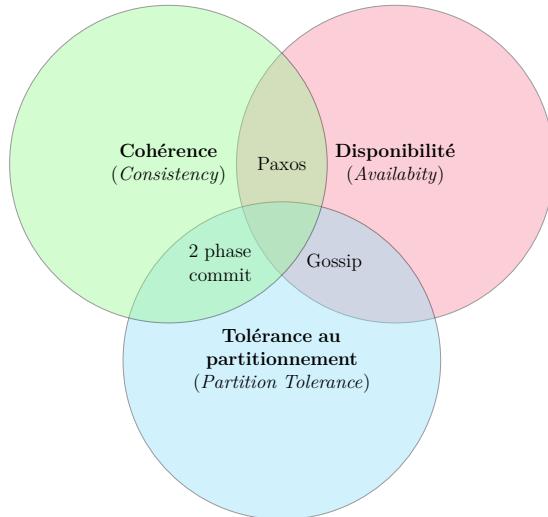


FIGURE 3.3 – Théorème CAP et les algorithmes de compromis

[Availability](#), and [Partition Tolerance \(CAP\)](#) (Figure 3.3). « *The largest single benefit about CQRS is when you start running into problems with the CAP theorem* » [102]. 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 découpée et subit un traitement réparti dont résulte la cohérence éventuelle de l'application. Dans la conception d'interaction pour l'interface, la synchronicité peut varier en fonction certains contextes. Les interactions asynchrones rajoutent souvent des étapes qui polluent l'interface. Par exemple, dans le cas de l'interaction suivante : **soumission de formulaire -> envoi asynchrone -> message de confirmation**, si l'utilisateur attend que la soumission soit une requête qui peut probablement échouer, alors l'asynchronisme se justifie pour être capable de fournir une explication à l'utilisateur en cas d'erreur. La probabilité d'échec peut être réduite en pré-validant la commande. Si personne d'autre que l'utilisateur ne travaille sur l'agrégat, la probabilité d'échec est quasi-nulle. De ce fait, l'asynchronicité rajoute une interaction inutile au flux dans le cas où peu de gens travaillent sur la même instance d'agrégat. Par exemple, il est intéressant de rendre l'exécution d'une commande synchrone et la mise à jour de la vue asynchrone. Les modèles de cohérence sont des décisions liées au métier car ils ont un impact direct sur l'expérience utilisateur.

3.2.4 Potentielles applications et autres utilisations

La conception et l'implémentation d'une plateforme comme 3DEvent qui est asynchrone, distribuée et orientée évènements (notamment la persistance) peut être appliquée à différents champs.

GIT-like app Les solutions pour faire de la gestion de version, comme MeshGit [23]

par exemple qui fait du *diff and merge* de maillages polygonaux pour des données 3D, sont rarement implémentées (a fortiori en temps-réel) sur des plateformes web à cause du coût et de la complexité que cela peut apporter dans des architectures traditionnelles. 3DEvent peut reconstruire n'importe quel état antérieur grâce à son architecture orientée évènements et indiquer les différences entre deux états.

Scenarii et *path recording* Pour des jeux sérieux, les graphistes 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 ou pour relever des habitudes (de travail) par exemple. Dans l'exemple du jeu sérieux, il est possible de comparer la trace utilisateur à la trace experte et permettre de rejouer le même scénario plusieurs fois facilement pour observer l'évolution. Ce type de fonctionnalité est intrinsèque à 3DEvent.

Traçage utilisateur et *crowdsourcing* 3DEvent peut se révéler être un bon outil pour enregistrer la trace d'un utilisateur lorsqu'il navigue dans la scène. L'enregistrement du chemin de la caméra et des actions de l'utilisateur sous la forme d'évènements 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 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 évènement est anormalement représenté dans le journal des évènements, il sera possible de lever une alerte facilement.

3.2.5 Bilan

Le modèle orienté évènement 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 évènements 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'évènements qui leurs sont associés. Cette étape introduit des différences fines sur les évènements qui sont produits à partir d'un agrégat. Cet aspect est notamment intéressant pour l'observation et la surveillance du contenu des sessions collaboratives développés en Section 4.2.3. Les composants du patron **CQRS** favorisent le découplage de l'application pour obtenir une gestion de la cohérence plus fine.

La partie **ES** s'intéresse à la sauvegarde des évènements dans un journal d'évènement, 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 évènements.

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 évènements pour la collaboration doit désormais être établie entre les collaborateurs.

3.3 Architecture de communication hybride

Dans la section précédente, l'introduction du modèle orienté évènements 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 et l'échange des données nécessaires à la collaboration : données de connexion, mises à jour de la scène, sensibilisation

attention
garder
le bon
endroit

... Afin de pouvoir proposer un **SEC** adapté aux besoins de l'édition de scènes 3D, plusieurs critères doivent être respectés :

- granularité fine pour l'édition massive ;
- spécifique à la modélisation 3D ;
- reposant sur des technologies web (réseau, interface, interaction 3D).

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** [86]. 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 conflit et de maintien de cohérence au cours des sessions de collaboration. Cela passe par la mise en place d'une architecture de communication hybride pour la collaboration. 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 contenu 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.

Cette section décrit le modèle d'architecture mis en place pour gérer la transmission de contenu entre les différents clients participant à l'édition collaborative d'un espace de travail 3D. Ces travaux 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.

Constat Les systèmes **P2P** qui sont orientés événements consistent en plusieurs nœuds interconnectés qui ont des fonctions similaires et exécutent des tâches similaires. Les pairs partagent directement leurs ressources (contenu, CPU, stockage, bande passante) sans nécessiter de serveur central. À la place, les pairs coopèrent aux moyens d'événements qui prennent la forme de messages échangés entre pairs. Les systèmes **P2P** sont capables de s'adapter aux dysfonctionnements et à la dynamité 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 contenu et pour les services d'intergiciel comme le routage et la localisation, l'anonymat et la confidentialité.

[WebRTC](#) est une technologie qui fournit aux navigateurs *destkop* et mobiles la possibilité de communiquer en temps-réel (cf [2.2.4](#) via une collection de standards, protocoles et [APIs](#) JavaScript. 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](#)). Techniquelement, [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).

Contribution Une architecture de communication hybride en trois parties serveur, persistance, pairs est utilisée dans 3DEvent [\[26\]](#). Cela 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...). Dans le contexte des [EVC 3D](#), le but est d'obtenir une architecture de communication :

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

Dans cette section, les différents composants de l'architecture sont détaillés. Ensuite sont expliqués les mécanismes de mise en relation de ces composants pour que les différents participants d'une session collaborative puissent se communiquer de manière transparente, cohérente et résistante aux pannes.

3.3.1 Présentation générale

Un intergiciel (*middleware*) se compose en général d'une [API](#) de recherche (Figure [3.4](#)). Dans notre modèle cette brique est principalement en lien avec le gestionnaire d'instance qui se charge de la recherche. L'[API](#) de messagerie concerne plus directement le système d'envoie et de réception de message. Les différents messages vont

concerner la partie routage de l'information, le maintien à jour du répertoire de voisin et leur dynamicité, ainsi que la description des connexions à maintenir. Le bloc de gestion de session permet au pair de savoir avec qui il est connecté et contient également les méta informations liées à ces liens (temps de connexion). Chaque pair doit également posséder un endroit où stocker les informations de contenu à traiter qui correspond au journal d'évènements. Le stockage peut prendre plusieurs formes : *in-memory*, stockage local, disque dur... selon le type de stockage adapté et disponible. Un pair est également au courant de son rôle. Le rôle d'un pair permet de distinguer s'il est un super pair, un pair normal, un pair actif, ou un pair passif... Le rôle induit les capacités et la configuration de chaque pair, i.e. un pair passif est configuré uniquement pour relayer les données tandis qu'un pair passif ne fait que stocker et relayer les données.

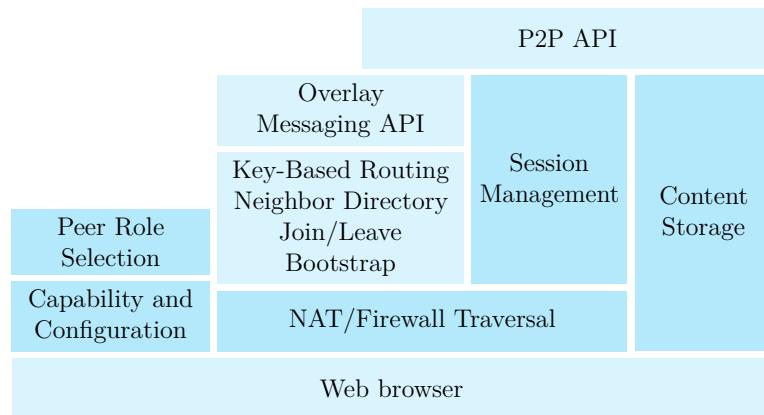


FIGURE 3.4 – Composants de chaque pair dans 3DEvent (point de vue réseau)

Le serveur assure d'une part la centralisation du stockage à long-terme et d'autre part la mise en relation des différents clients.

La couche P2P fournit une dissémination rapide de l'information et décentralise la réPLICATION des données à court terme. Cela évite donc au serveur d'être le point central des échanges en déchargeant les canaux passant par le serveur.

Connexion des pairs en début de session La Figure 3.5 représente la séquence d'actions nécessaire à une instance 3DEvent (*idA*) pour rejoindre le réseau contenant déjà d'autres instances 3DEvent. L'action *join* est exécutée lorsqu'un utilisateur envoie ses informations de connexion sur un portail de connexion (à partir d'une instance web) ou lorsqu'une instance serveur est lancée. Cette action ajoute le nouveau pair à la liste des pairs présents sur le réseau. La liste est gérée par le gestionnaire d'instance, et retourne la liste des pairs avec lesquels le pair doit se connecter. Pour

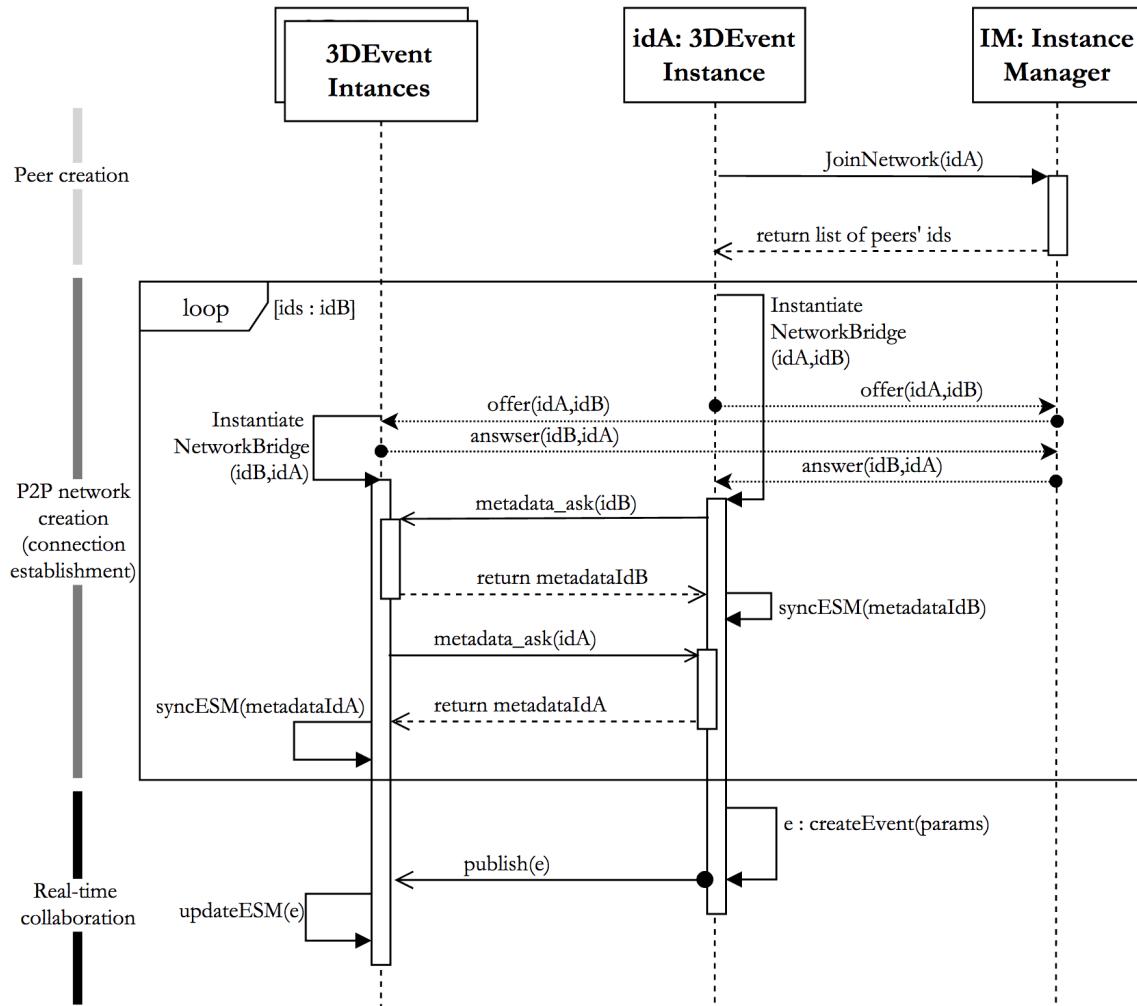


FIGURE 3.5 – Protocole de connexion au réseau d’instance 3DEvent

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 *network bridge* dans l’*event store* de idA puis celui de idB . Afin de resynchroniser les deux pairs (après cette série d’échanges asynchrones), idA et idB s’échangent des métadonnées sur la situation respective de leurs *ESM* afin de se synchroniser .

3.3.2 Extension : Event Store distribué

L’Event Store est un composant clé dans le traitement des évènements. Il prend en entrée des évènements générés ou reçus extérieurement et produit en sortie des évènements cohérents qui peuvent être par la suite publiés. Les évènements sont considérés comme cohérents lorsqu’il n’y a pas d’erreur de cohérence, i.e. lorsque les numéros de versions sont bien ordonnés. Chaque Event Store contient deux éléments : l’Event Stream Manager (ESM) et des Network Bridges (NB). Un ESM

est une structure de données représentée par une collection de flux d'évènements. Un flux d'évènement est représenté par un tableau d'évènements indexés en séquence permettant de stocker les évènements d'un agrégat dans l'ordre temporel. Si un flux 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 évènement de l'instance courante, l'ESM récupère (ou crée) le flux d'évènements associé à l'agrégat référencé par l'évènement. Alors, la cohérence de la version est vérifiée en comparant la version attendue (exposée dans les métadonnées de l'évènement) et la version actuelle de l'agrégat. Si les deux numéros de versions sont égaux, l'évènement est ajouté à la fin du tableau du flux pour être stocké, sinon une exception est levée. La gestion des exceptions est expliquée dans . Une fois le stockage effectué dans l'ESM, l'évènement est publié.

3.3.3 Persistance à long terme

Dans un contexte industriel de collaboration, l'information doit être disponible sur le long terme, facilement accessible par l'entreprise. La persistance à long-terme stocke le journal d'évènement 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.

3.3.4 Synchronisation client-serveur

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 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 d'autres pairs sans le serveur) : dans le cas où le serveur est absent, les clients peuvent continuer de créer en collaborant. Ces données n'étant stockées que sur le client, il est nécessaire de les transmettre à la base de données dès qu'une connexion est possible. Cela peut être fait en une fois ou de manière partagée ;

- (c) travail "online" (en collaboration avec d'autres pairs avec le serveur) : le client envoie régulièrement ses nouvelles modifications pour qu'elles soient intégrées à la base de données.
2. depuis la base de données : Le client reçoit toutes les nouvelles mises à jour de la scène depuis la dernière fois qu'il s'est connecté. Cela peut également inclure des mises à jour qui sont en conflit avec ce qu'il y a dans son propre espace de stockage qu'il lui faut donc modifier. Dans le cas où un utilisateur est seul connecté, la base de données est la seule source disponible pour la mise à jour du client.

3.3.5 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. Chaque gestion de commande (Figure ??) entraîne la génération d'un ou plusieurs événements. Ces événements sont considérés comme « soumis » (*uncommitted*) mais pas encore « publiés » (*committed*). Pour qu'ils deviennent, l'agrégat concerné par ces événements doit produire une nouvelle version sans être en conflit avec la précédente. En passant la version attendue v_a au gestionnaire de conflit, on est à même de la comparer avec la version courante v_c . Il existe deux cas où un conflit est levé :

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

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

3.3.6 Gestion de la cohérence

Respect de la causalité

Convergence des répliques

Préservation de l'intention

3.3.7 Bilan

3.4 Conclusion du chapitre

Dans ce chapitre nous avons vu les différents composants de l'architecture de communication pour la réalisation d'un **EVC 3D**. En mettant en avant la technologie WebRTC, nous avons montré qu'il était possible de réaliser une architecture qui respecte les standards du web et l'interopérabilité nécessaire à ce type d'environnement. La mise en place d'une architecture décentralisée dans un environnement distribué permet de mettre à contribution tous les acteurs de la collaboration. De ce fait, l'accessibilité des ressources est renforcée par la présence de nombreuses unités présentes sur le réseaux. Cela permet à la fois de récupérer du contenu rapidement et d'octroyer une autonomie certaine aux contributeurs.

Chapitre 4

Implantation

Contents

4.1	Introduction	74
4.2	3DEvent : Plateforme web de manipulation collaborative d'objets 3D	74
4.2.1	Éditeur 3DEvent	74
4.2.2	Interface utilisateur	75
4.2.3	Flexibilité de la visualisation	78
4.3	Intergiciel P2P pour l'échange de données 3D	78
4.3.1	Données d'échange	80
4.3.2	Synchronisation des données	81
4.4	Résumé des choix techniques	83
4.5	Bilan	84

4.1 Introduction

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

Cette section présente l'implantation de 3DEvent, la plateforme web de manipulation et visualisation collaborative d'objets 3D réalisée à partir du modèle présenté dans le chapitre précédent. La première partie s'intéresse à l'intégration du framework pour proposer une application d'assemblage d'objets 3D. La seconde partie expose les choix techniques pour l'interface utilisateur et la mise en avant du système de visualisation flexible.

4.2.1 Éditeur 3DEvent

3DEvent est à la fois un **framework** et un éditeur 3D pour la visualisation et la manipulation d'objets 3D. La partie **framework** est basée évènement pour répondre à des contraintes liés à la temporalité de l'information traitée via l'éditeur ainsi qu'à la distribution de l'information en terme d'intégrité et de poids. L'éditeur va réagir et interpréter les informations (évènements) fournies par le **framework** pour proposer visualisation et interactions adaptées.

L'application 3DEvent est un éditeur 3D reposant sur les principes et les technologies du web qui permet de manipuler des objets 3D de manière collaborative en temps-réel. Les interactions possibles sont :

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

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

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

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

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

L'intérêt de proposer une application web se retrouve principalement dans l'accessibilité qu'elle propose. En effet, n'importe quel terminal muni d'un navigateur web peut y accéder, ce qui la rend distribuée et multi-plateforme.

Les fonctionnalités graphiques proposées par WebGL sont un peu réduites par rapport à celles d'OpenGL dont l'[API](#) évolue plus vite et propose plus de flexibilité et d'optimisations. Cependant, les performances graphique restent très correctes car le navigateur est quand même capable d'utiliser les processeurs graphiques du terminal (GPU) pour les calculs et rendus 3D.

4.2.2 Interface utilisateur

Dans le but de proposer une [IU](#) proche des fonctionnalités métiers liées à la modélisation 3D avec une interface orientée tâche. De cette manière, le modèle orienté événements présenté dans le précédent chapitre est directement orienté métier.

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 lui permettant d'utiliser différentes fonctionnalités. Les deux modalités d'interaction sont le clavier et la souris. Le premier niveau de cette interface est scindée en deux panneaux :

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

L'onglet "Scène" possède un bloc contenant les détails d'un maillage en cours de sélection. Cela permet d'avoir la description des propriétés de l'objet sélectionné et une manipulation de ses paramètres (position, rotation et mise à l'échelle) plus précise que via l'espace 3D avec le cliqué/déplacé .

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

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

Dans chaque onglet on trouve donc différent blocs **HTML**, avec des comportements spécifiques à un agrégat et injectés dynamiquement. Ces blocs correspondent aux Views de notre modèle.

Exemple d'interaction dans le framework La Figure 4.1 décrit la façon dont le système traite l'exécution d'une commande de translation déclenchée par l'utilisateur et comment cette information est diffusée à ces collaborateurs².

ref a la fig
et traduc-
tion. cote
implé ?

Le composant responsable de la détection de conflit permet au développeur d'implémenter ses propres règles de résolution de conflit. Le composant déclenche une exception lorsque le numéro de version reçu et le numéro de version courant de l'agrégat sont identique (Figure 4.1 étape (c)). Selon les règles métiers définies et les exceptions liés à la cohérence, l'événement peut être business logic rules defined, the event is rejected, accepted with or without modifications. From this process, new events can be generated during the resolution.

Figure 4 describes how the system executes a translation action triggered by a user and how it's broadcasted to his/her collaborators (scene, cube geometry, and cube mesh should be created before).

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

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

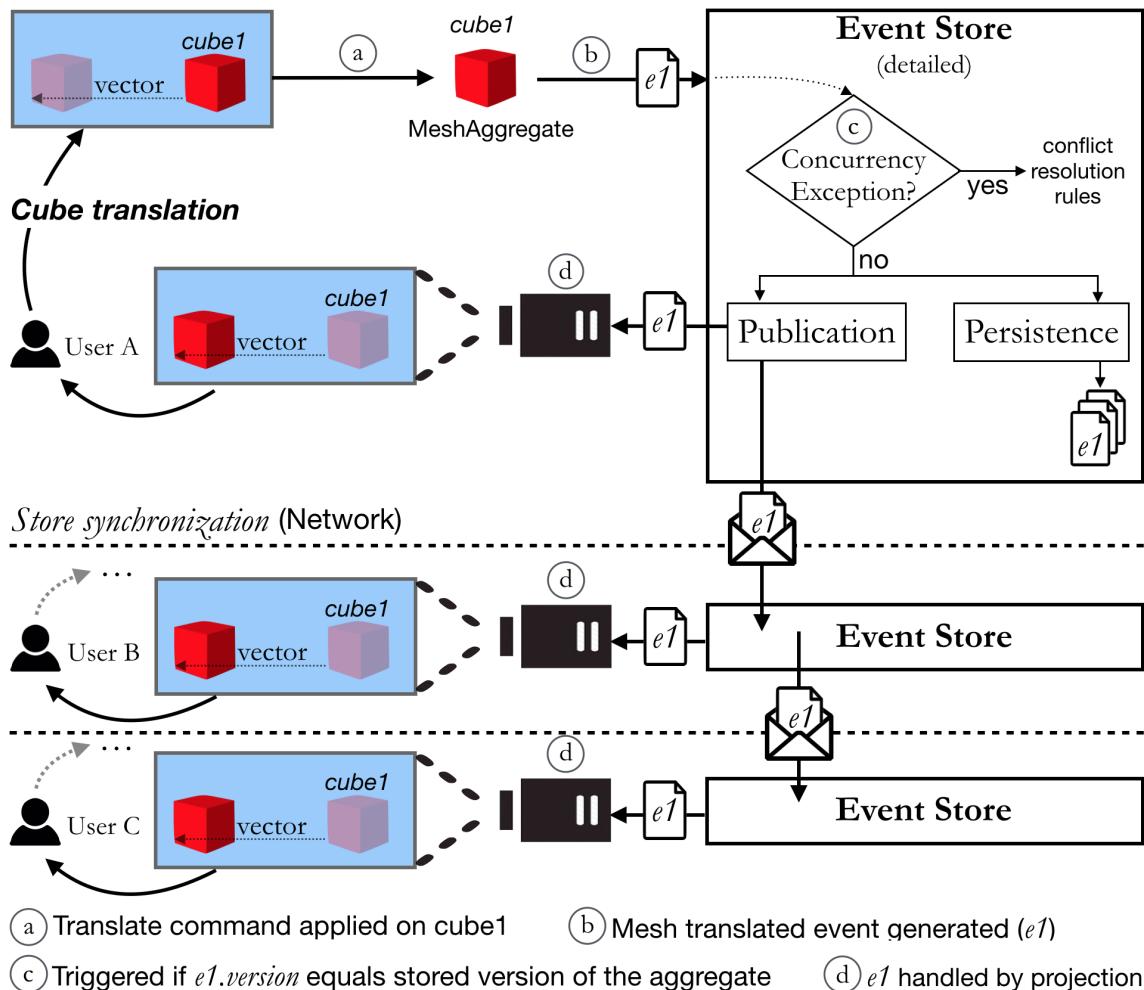


FIGURE 4.1 – Exemple d'édition collaborative où User A est connecté à User B, lui-même connecté à User C. Le cycle montre les différentes étapes du déclenchement de la commande au rendu visuel en passant par la génération de l'évènement, la synchronisation du journal d'évènements et l'impact sur le rendu des autres utilisateurs pour une translation sur un cube.

4.2.3 Flexibilité de la visualisation

Dans l'approche **CQRS**, une projection est une dérivation de l'état courant à partir du flux d'évènements. Pour Abdullin, «la projection est le processus de conversion (ou d'agrégation) d'un flux d'évènement en une représentation structurelle. Cette dernière (qui est mise à au moment où le flux est parcourue) peut être avoir différentes appellations : modèle de lecture persistant, vue ou état.» La partie lecture du modèle (l'affichage sur interface utilisateur) bénéficie des projections en lui permettant de réduire l'afflux des évènements, ne laissant filtrer que ceux qui sont pertinents pour la vue. La projection fournit une vue adaptée (filtrée, enrichie...) du flux d'évènements 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 évènements ou de manière asynchrone et donc découpée du flux des évènements.

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 pouvant arriver dans n'importe quel ordre, qu'arriverait-il si un évènement A (eA) nécessitant un autre évènement B (eB) pour être appliqué arrivait avant ? Dans cette situation, le système va générer une erreur en essayant d'appliquer eA sur un état inadéquat car il n'a pas d'information sur la hiérarchie d'application des évènements (eB puis eA).

Pour pallier 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 et qui réalisent les actions déterminées en fonction des évènements qui arrivent. Par exemple, si on essaie d'ajouter un objet dans une scène (eA) sans avoir créer 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 .

4.3 Intergiciel P2P pour l'échange de données 3D

L'architecture de communication décrite dans le chapitre précédent nécessite l'implantation d'un intergiciel P2P compatible avec les besoins liés à la 3D, le web et la collaboration.

L'assumption est faite que tous les clients utilisent des navigateurs qui implémentent et supportent le protocole WebSocket et l'**API RTCDataChannel**.

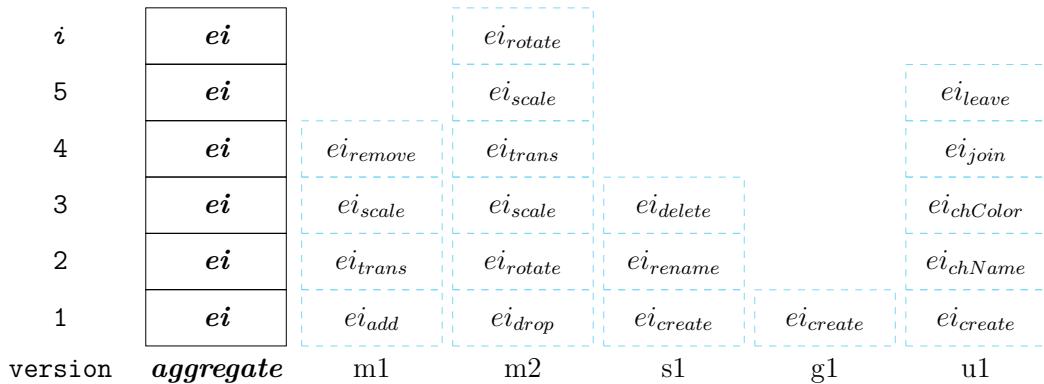


FIGURE 4.2 – Structure d'un agrégat

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

La topologie de l'architecture de communication repose sur la mise en relation automatique des clients par le biais du serveur pour établir une connexion [WebRTC](#). Pour ce faire, chaque client envoie son identifiant (ID) lors de sa première requête au serveur qui le stocke. Selon le paramétrage de la connectivité directe minimum établie préalablement, le serveur recherche aléatoirement l'ID d'autres clients qui satisfont la règle de connectivité. Cette règle de connectivité minimum permet d'ajuster la densité du maillage (connectivité élevée : maillage partiel dense voire complet ; connectivité faible : maillage partiel épars) 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 « rebondir » pour atteindre tous les pairs et par conséquent le temps de transmission est augmenté (exemple d'une distribution en ligne).

De navigateur à serveur

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

De navigateur à navigateur

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

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

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

4.3.1 Données d'échange

En sachant que le modèle est conçu pour des applications web, 3DEvent a besoin d'un format de données permettant de faire communiquer des acteurs hétérogènes du système. Le format [JavaScript Object Notation \(JSON\)](#), dérivé de la notation des objets du langage JavaScript, est lisible et interopérable. Le Listing 4.1 montre un exemple d'événement en format JSON. Ce type de données est assez abstrait et suffisamment générique pour représenter n'importe quelle donnée. Par exemple le format de fichier [GL Transmission Format \(glTF\)](#) se base sur la représentation [JSON](#) afin de décrire une scène 3D. Le format [JSON](#) est aussi utilisé pour la sérialisation et la désérialisation des objets transmis par RTCDataChannel qui prend n'importe quel format de données.

L'[API](#) RTCDataChannel supporte beaucoup de types de données différents (chaînes de caractères, types binaires : Blob, ArrayBuffer...). Dans un environnement multi-utilisateur avec des données hétérogènes (3D, images, informations) tel que 3DEvent cela facilite l'interopérabilité.

Listing 4.1 – Mesh added to Scene event and parameters

```
1 {
2 {
3 "sceneId": "scene-turbine",
4 "meshId": "406514c6-306b-f0f9643a037e",
```

```

5 "geometryId": "37076875-ea1c-bbd300481345",
6 "name": "blade",
7 "color": "#963912"
8 },
9 "version": 17,
10 "author": "Foo"
11 }

```

4.3.2 Synchronisation des données

Persistance à court terme

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

Ce stockage fonctionne sur un système de clé/valeur qui rend facile l'accès aux événements enregistrés sur le client. La configuration du client est également stockée localement.

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 [21, 43], progressif [19, 60], mise en cache [44], compression / décompression [56, 76, 63]. Une catégorisation plus précise du flux continu peut être donnée selon quand le contenu est généré : en direct (*live* ou *push*) et à la demande (*on-demand* ou *pull*).

Le mécanisme de routage que nous avons utilisé dans [27] est proche du routage de GNutella.

Gestion des évènements des agrégats

La configuration des connexions RTCDataChannel peut être configurée de selon les critères (Section [ref section config](#)) de fiabilité et d'ordonnancement. Dans l'intericiel de 3DEvent, cette configuration est : non fiable et non ordonnée. C'est l'application qui est en charge de « ré-ordonner » les évènements. En effet, les évènements sont associés à des agrégats. Comme chaque agrégat possède un flux d'évènements dédié et numéroté il est alors simple de les ordonner lorsque. Lors de la synchronisation (Section [ref section sync](#)), les évènements manquants sont demandés successivement à tous les pairs. Lors qu'un pair reçoit les évènements il les stocke dans le flux correspondant à l'agrégat. Si un évènement est manquant, les suivants sont quand même stockés en laissant l'espace de l'évènement manquant dans le tableau. L'évènement est redemandé jusqu'à son obtention, laquelle provoque le déblocage de l'état de l'agrégat jusqu'au prochain évènement manquant (ou la fin du flux). Les évènements 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 [ref section](#)) qui réduisent la taille des flux en mémoire pour chaque agrégat qui part déjà d'un état avancé.

Sélection fantôme

Les interactions utilisateurs doivent être adaptées à la collaboration et aux manipulation 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) (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 . En différenciant l'actuel objet que l'utilisateur souhaite sélectionné (O_o) de celui manipulé (O_f), plusieurs aspects de l'interaction sont mis en valeur pour :

- 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 (faire l'action puis la défaire) inutiles 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 car 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'évènements comme lorsque l'utilisateur change d'idée lors de l'interaction ou suite à une intervention concurrente. L'utilisateur n'a un impact sur l'application que lorsqu'une modification métier est réalisée.
- le réseau : l'information importante à faire transité est l'évènement correspondant à la modification métier pas toutes les positions intermédiaires même si intuitivement l'idée de temps réel pourrait conduire à cette solution. La quantité de messages produite surchargerait à la fois le réseau et le fil d'exécution principale de l'application. En effet, WebRTC a l'inconvénient pour le moment de ne 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.

Lorsque l'utilisateur relâche O_f , alors la modification intentée s'applique sur O_o avec le principe du *Last Write Last Win* (le dernier gagne).

4.4 Résumé des choix techniques

Base de données NoSQL L'évolution de l'utilisation du web en tant que plate-forme applicative a encouragé le changement dans le stockage des données pour de nouveaux besoins supportant de larges volumes de données (comme les données 3D). Une base de données **Not Only SQL (NoSQL)** fournit un schéma libre et dynamique ainsi qu'une API de requête riche pour la manipulation de données. De plus, la possibilité d'enrichir un document à la volée facilite l'évolution des objets (3D) et la

maintenance. de l'application. 3DEvent intègre un système de persistance sur le long terme caractérisé par une base de données [NoSQL](#).

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

4.5 Bilan

Chapitre 5

Expérimentations

Contents

5.1	Introduction	86
5.2	Cas d'étude : Assemblage collaboratif d'objets 3D dans un environnement web	86
5.3	Expérimentation 1 : preuve de faisabilité	86
5.3.1	Présentation de l'expérimentation	86
5.3.2	Résultats	86
5.3.3	Discussion et Conclusion	86
5.4	Expérimentation 2 : Intégration du framework événementiel	86
5.4.1	Résultats et Discussion	89
5.5	Comparaison entre l'expérimentation 1 et l'expérimentation 2	91
5.5.1	Résultats	91
5.5.2	Discussion et Conclusion	91
5.6	Bilan	91

5.1 Introduction

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

Dans [27] et [24],

5.3 Expérimentation 1 : preuve de faisabilité

Cette expérimentation est tirée de l'article [27]. 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.

5.3.1 Présentation de l'expérimentation

5.3.2 Résultats

5.3.3 Discussion et Conclusion

5.4 Expérimentation 2 : Intégration du framework évènementiel

L'expérimentation propose de répliquer une collaboration réaliste entre différents participants travaillant à distance. Ce cas d'étude présenté dans [24] souligne plusieurs aspects relatif à la fiabilité du modèle et de son implantation de deux manières :

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

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

Tâche à effectuer Les participants devaient assembler les différentes parties d'un modèle 3D en utilisant l'application 3DEvent et les fonctionnalités décrites précédemment pour que le résultat corresponde à l'assemblage donné en exemple

(images). La complexité de la tâche a été modulée selon plusieurs facteurs : le nombre de parties composant le modèle et le nombre de collaborateurs. Afin de permettre la manipulation des objets 3D facile à apprendre et utiliser, il a été choisi de conserver des manipulations de haut niveau.

Population L’expérience a été conduite sur 12 groupes de deux ou trois participants chacun. Chaque participant était localisé en France dans une zone urbaine avec de bonnes infrastructures en travaillant sur des réseaux distincts avec une bonne connexion internet (au moins 20Mb/s) pour éviter d’avoir des latences extrêmes (>10 secondes) au cours des expérimentations. Les participants étaient des étudiants de Master ou de Doctorat en informatique (pas nécessairement familiers avec les environnements 3D). Les participants étaient autorisés à communiquer entre eux par chat.

Procédure Les modèles utilisés lors de l’expériences sont décrits dans le Tableau 5.1. L’expérimentation se déroule en trois phases :

Phase d’essai Chaque participant s’entraîne pendant 5-10 minutes sur un modèle de test dans l’application pour se familiariser avec l’interface et les fonctionnalités proposées.

Phase solo Le participant effectue un assemblage du modèle Rotor ou Camera Box.

Phase de collaboration Un groupe de participants réalise deux assemblages sur (i) un petit modèle (10 ou 12 parties) puis (ii) un plus gros modèle (16 parties). La phase de collaboration a été répétée six fois : trois fois avec un groupe de deux participants et trois fois avec un groupe de trois participants.

Du fait que 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.

Tableau 5.1 – Modèles utilisés durant l’expérimentation

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

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

Données collectées Pour chaque expérimentation et chaque participant, le temps de complètement, le nombre d'événements générés et l'horodatage de chaque événement pour observer le complètement de la tâche en termes 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 évènement sur la scène initialisée est généré (horodatage du premier évènement) ; et il s'arrête lorsque le groupe indique que la tâche à compléter est terminée (horodatage du dernier évènement est considéré).

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.3) .

Le questionnaire est inspiré de [57], qui permet d'évaluer la facilité d'utilisation du système et l'implication dans la collaboration de chacun des participants. En utilisant une échelle de notation sur sept points [58], (1 : pas d'accord ; 7 : d'accord) pour avoir assez de points de discrimination.

Au cours des différentes sessions collaboratives, l'application à produit plusieurs centaines d'évènements (environ 300 par session). Les expérimentations ont été réalisées sur plusieurs dispositifs notamment un *smartphone* avec une connexion 4G. La Figure 5.1 montre quelques captures d'écran durant une session collaborative sur le modèle Rotor ; et la Figure 5.2 montre les premiers évènements enregistrés dans la base de données. Les boîtes englobantes représentent la sélection des différents collaborateurs pendant la session.

5.4.1 Résultats et Discussion

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

Figure 5.3 montre deux angles d’enregistrement d’une session sur le modèle *living room*. Au début de la session, beaucoup d’objets sont ajoutés (meshAddedToSceneEvent). Seul un utilisateur a ajouté un objet en utilisant la fonctionnalité pour déposer un objet à un endroit spécifique de la scène (meshDropped). Le nombre d’évènements concernant la sélection et la désélection est à peu près similaire aux exception. La différence s’explique par le fait que l’évènement de désélection n’est pas déclenché lors d’un changement d’objet sélectionné, car la désélection n’est pas effectuée explicitement par l’utilisateur. Au commencement, les participants ont fortement interagis (jusqu’à 20 évènements en 15 secondes). Cela s’explique par le recours massif à l’ajout d’objets dans la scène pour composer le modèle et la mise à l’échelle de ceux-ci (du au positionnement arbitraire des objets de la bibliothèque). Ensuite, les trois utilisateurs interagissent ensemble durant quelques minutes avant que l’un d’entre eux ne quitte et ne reviennent quelques secondes plus tard (informations récupérées à partir du journal d’évènements lié aux agrégats utilisateurs). Enfin, le nombre d’évènements diminue montrant la fin de l’activité, les utilisateurs finissant la tâche et ajustant les derniers objets.

L’implication d’un utilisateur peut être vue par le prisme de la fréquence de ses contributions et la variété de fonctionnalités utilisées, autrement dit, les différents types d’évènement 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’évènement (ou d’un motif d’évènements répété) peut montrer une faiblesse de l’ergonomie de l’interface. Pour ce dernier aspect, on peut prendre l’exemple dans la Figure 5.3b à 45s, on remarque un nombre élevé de MeshScaledEvent. A

posteriori, nous nous sommes rendu compte que la fonctionnalité était mal calibrée pour l'échelle du modèle et nécessitait donc de s'y reprendre à de nombreuse fois pour réaliser la bonne transformation.

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

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

Dans toutes les expérimentations menées, le but a été atteint dans pratiquement le même temps (10-15 minutes). La facilité d'utilisation du système est bien notée mais indique encore quelques aspects à améliorer. Pour modérer ces résultats, il est important de rappeler que bien que la complexité des modèles n'est pas extrême, tous les participants étaient débutants sur le système et n'était pas forcément familier de ce genre d'application. Le fait que l'application soit basée 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 medium (3D) inhabituel pour ce genre de plateforme. De plus, on peut également supposer que le prototype créé pour l'expérimentation correspond bien au

à l'objectif d'assemblage coopératif d'objets 3D vu que les tâches ont rapidement été réalisées. Sur une échelle de « non-interactif » à « temps-réel » les participants ont qualifié l'application comme « quasi temp-réel ».

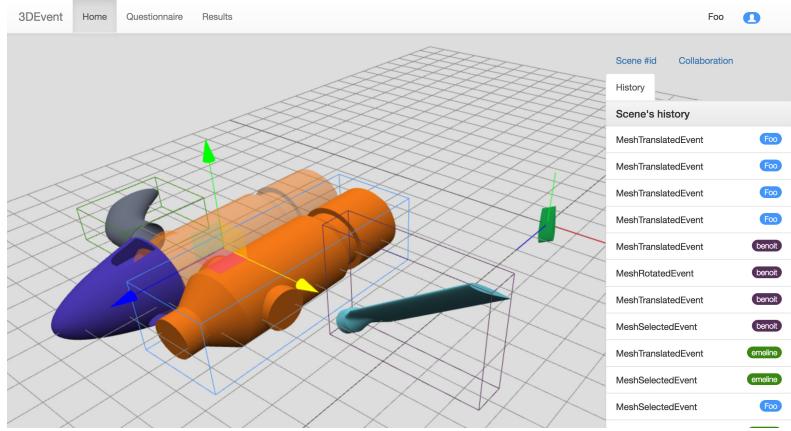
La satisfaction générale à propos de l'expérimentation et la satisfaction concernant la collaboration et l'expérience utilisateur montrent que les participants ont positivement apprécié faire de la modélisation collaborative 3D dans un navigateur web. Quant au fait que le nombre d'utilisateur améliore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

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

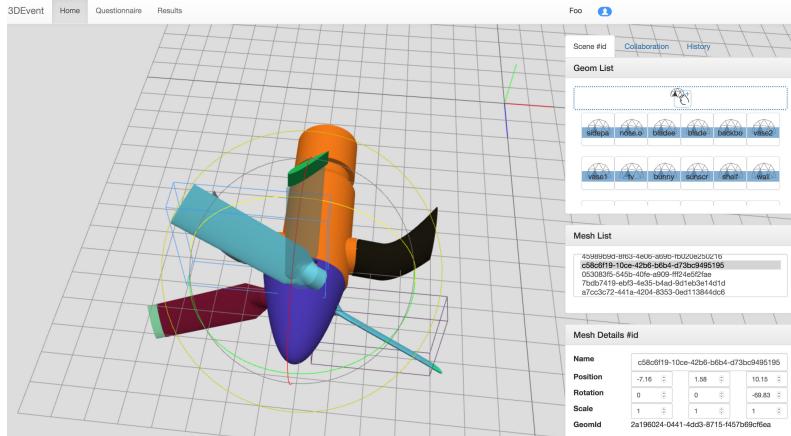
5.5.1 Résultats

5.5.2 Discussion et Conclusion

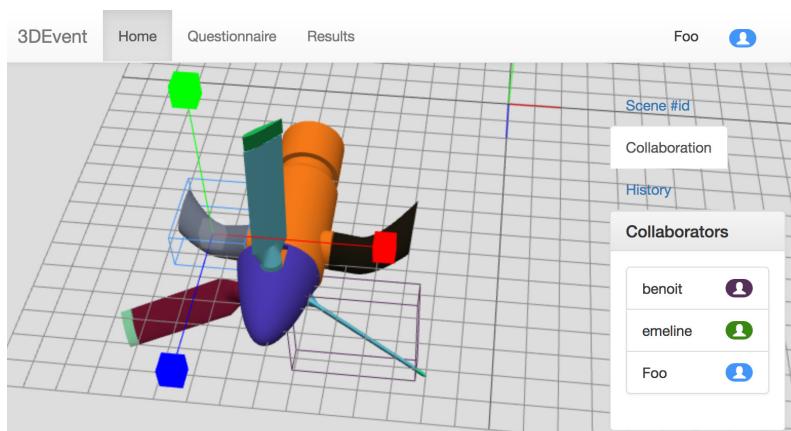
5.6 Bilan



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

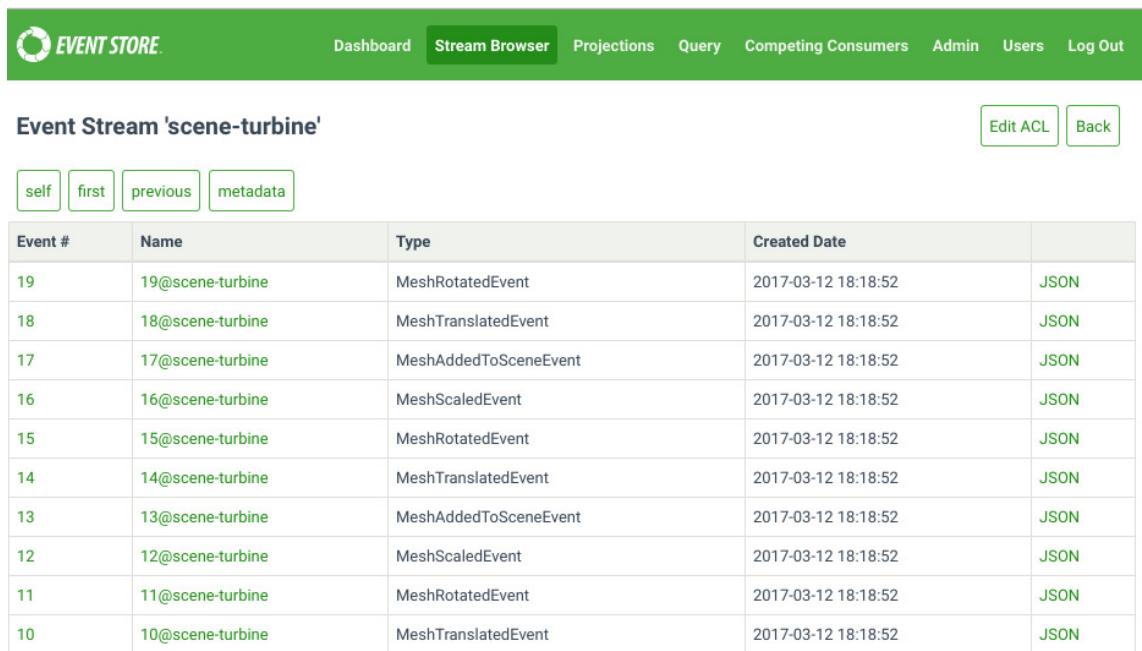


(b) Rotation (environnement 3D) et outils pour la manipulation d'objet 3D (panneau latéral)



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

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



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

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

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

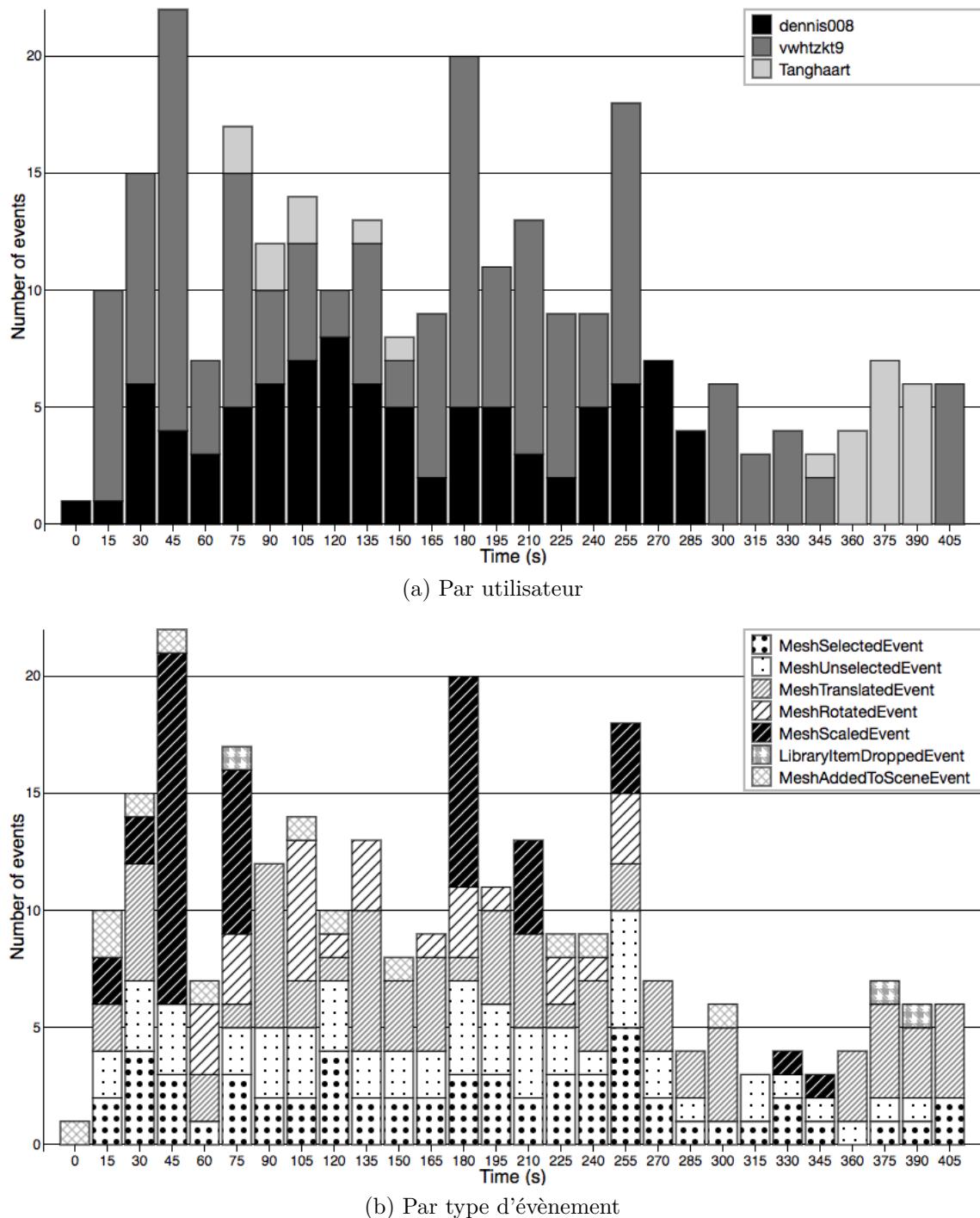


FIGURE 5.3 – Résumé d'une session collaborative au cours du temps

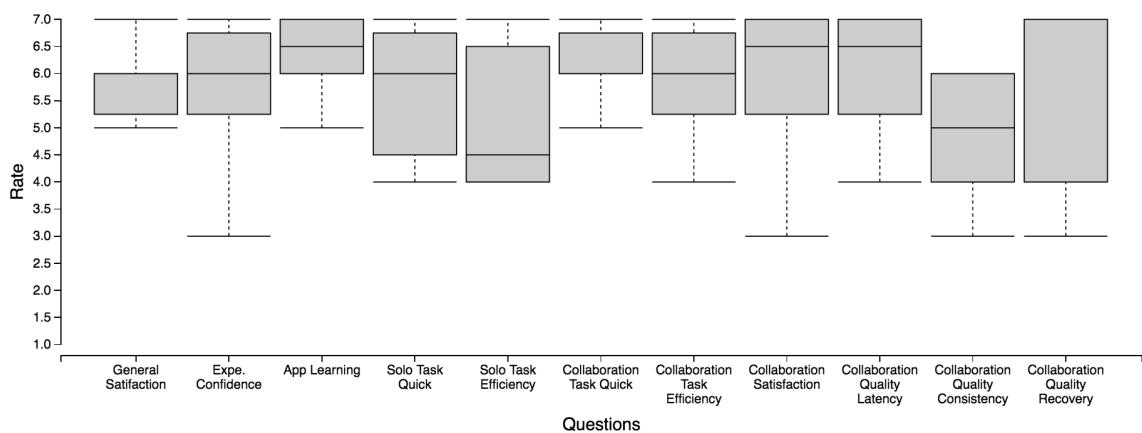


FIGURE 5.4 – Résultats des questionnaires collectés

Chapitre 6

Conclusion

Cette thèse présente une architecture évènementielle pour les **EVC** sur le web. Les contributions de cette thèse sont multiples. Il y a d'une part les contributions scientifiques qui porte sur l'**EDA** pour la modélisation 3D collaborative et d'autre part, il y a la présentation d'une architecture de communication hybride permettant de synchroniser les **EVCs 3D** sur le web de différents client.

Chacune de ces contributions est accompagnée de son implantation.

De plus, la réalisation de plusieurs prototypes fonctionnant sur différents paradigme pour les expérimentation ont permis de montrer les avantages et inconvénients de ceux-ci.

Enfin, les expérimentations principalement portées sur l'étude des utilisateurs permet de souligner plusieurs aspects du modèle. Le premier est de montrer que l'intégration du métier a permis l'observation minutieuse du travail réalisé au sein de l'environnement.

Annexe A

Ressources pour l'implantation et les expérimentations

A.1 Description des évènements par agrégat dans 3DEvent

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

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

A.2 Messages réseaux pour la synchronisation des Event Stores

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

Message	Description
STREAM_SYNC_ASK	Demande de synchronisation d'un <i>stream</i>
CHUNK	Réception d'une donnée <i>chunk</i>)
READY_ASK	Prêt pour la démarrer la demande de données de sync.
READY	Prêt pour démarrer la réception de données de sync.
ALL_EVENTS_SYNC_ASK	Demande de toutes les données typées évènement
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 évènement
END_SYNC	Fin de la synchronisation

parler des tableaux

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 Expérimentation 2 : User study questionnaire

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

Bibliographie

- [1] A. AVRAM et F. MARINESCU : *Domain-Driven Design Vite fait.* 2006. [2.6](#)
- [2] R. BALDONI, S. a. RUBERTI, L. QUERZONI et S. TUCCI-PIERGIOVANNI : TERA : Topic-based Event Routing for peer-to-peer. *International Conference on Distributed Event-Based Systems (DEBS)*, p. 2–13, 2007. [2.3.2](#)
- [3] G. BANAVAR, T. CHANDRA, B. MUKHERJEE, J. NAGARAJARAO, R. STROM et D. STURMAN : An efficient multicast protocol for content-based publish-subscribe systems. *Proceedings. 19th IEEE International Conference on Distributed Computing Systems (Cat. No.99CB37003)*, p. 262–272, 1999. [2.3.2](#)
- [4] J. Y. BANG, D. POPESCU, G. EDWARDS, N. MEDVIDOVIC, N. KULKARNI, G. M. RAMA et S. PADMANABHUNI : CoDesign : a highly extensible collaborative software modeling framework. *2010 ACM/IEEE 32nd International Conference on Software Engineering*, 2:243–246, 2010. [2.3.1](#)
- [5] I. BARAN : Under the Hood : How Collaboration Works in Onshape, 2015. [2.1.2](#), [2.2](#)
- [6] M. BECHER : Interactive Volume Visualization with WebGL. (06), 2012. [2.1.3](#)
- [7] J. BEHR, Y. JUNG, J. KEIL, T. DREVENSEK, M. ZÖLLNER, P. ESCHLER et D. W. FELLNER : A Scalable Architecture for the HTML5/X3D Integration Model X3DOM. *Proceedings of the 15th International Conference on Web 3D Technology*, 1(212):185–194, 2010. [2.1.3](#)
- [8] P. BENTLEY et J. WAKEFIELD : Generic Evolutionary Design. *Soft Computing in Engineering Design* . . . , p. 1–10, 1997. [1.1.4](#)
- [9] K. BIRMAN et T. JOSEPH : Exploiting virtual synchrony in distributed systems. *ACM SIGOPS Operating Systems Review*, 21(5):123–138, 1987. [2.3.2](#)
- [10] D. BROWN, S. JULIER, Y. BAILLOT et M. LIVINGSTON : An event-based data distribution mechanism for collaborative mobile augmented reality and virtual environments. *IEEE Virtual Reality, 2003. Proceedings.*, 2003, 2003. [1.1.4](#)
- [11] C. CALABRESE, G. SALVATI, M. TARINI et F. PELLACINI : cSculpt : a system for collaborative sculpting. *ACM Transactions on Graphics*, 35(4):1–8, 2016. [2.2](#)
- [12] S. CALLAHAN, M. SCHENK et N. WHITE : Building a collaborative workplace. *Anecdote : putting stories to work*, p. 1–11, 2008. [1.1.1](#)
- [13] A. CARZANIGA, D. S. ROSENBLUM, C. SCIENCE et A. L. WOLF : Achieving Scalability and Expressiveness in an Internet-Scale Event Notification Service. *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, p. 219–227, 2000. [2.3.2](#)

- [14] A. CARZANIGA et A. L. WOLF : A Benchmark Suite for Distributed Publish / Subscribe Systems. *Program*, 2002. [2.3.2](#)
- [15] M. CASTRO, P. DRUSCHEL, A. M. KERMARREC et A. I. T. ROWSTRON : Scribe : A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, 20(8):1489–1499, 2002. [2.3.2](#)
- [16] S. K. CHANDRASEGARAN, K. RAMANI, R. D. SRIRAM, I. HORVÁTH, A. BERNARD, R. F. HARIK et W. GAO : The evolution, challenges, and future of knowledge representation in product design systems. *Computer-Aided Design*, 45(2):204–228, 2013. [1.1](#)
- [17] M. K. CHANDY, O. ETZION et R. V. AMMON : *The event processing manifesto*. Num. 10201. 2011. [1.1.4](#), [3.2](#)
- [18] H.-M. CHEN et C.-C. HOU : Asynchronous online collaboration in BIM generation using hybrid client-server and P2P network. *Automation in Construction*, 45:72–85, 2014. [2.2.5](#)
- [19] W. CHENG, D. LIU et W. T. OOI : Peer-assisted view-dependent progressive mesh streaming. *Proceedings of the seventeen ACM international conference on Multimedia - MM '09*, p. 441, 2009. [4.3.2](#)
- [20] Y. CHENG, F. HE, X. CAI et D. ZHANG : A group Undo/Redo method in 3D collaborative modeling systems with performance evaluation. *Journal of Network and Computer Applications*, 36(6):1512–1522, 2013. [2.3.5](#)
- [21] C. H. CHU et Y. H. CHAN : Optimized product data transmission in LOD-based collaborative design using P2P CAD streaming. *Journal of Intelligent Manufacturing*, 23(5):1559–1571, 2012. [4.3.2](#)
- [22] V. CRISTEA, F. POP, C. DOBRE et A. COSTAN : Distributed architectures for event-based systems. *Studies in Computational Intelligence*, 347:11–45, 2011. [1.1.4](#)
- [23] J. D. DENNING et F. PELLACINI : MeshGit. *ACM Transactions on Graphics*, 32(4):1, 7 2013. [3.2.4](#)
- [24] C. DESPRAT, B. CAUDESAYGUES, H. LUGA et J.-P. JESSEL : Doctoral Symposium : Loosely Coupled Approach for Web-Based Collaborative 3D Design. In *Proceedings of ACM International Conference on Distributed Event-Based Systems*, 2017. [3.1](#), [3.2](#), [5.2](#), [5.4](#)
- [25] C. DESPRAT, J.-P. JESSEL et H. LUGA : A 3D collaborative editor using WebGL and WebRTC. *Proceedings of the 20th International Conference on 3D Web Technology - Web3D '15*, p. 157–158, 2015. [3.2.1](#)
- [26] C. DESPRAT, J.-P. JESSEL et H. LUGA : 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*, p. 73–76, 2016. [3.1](#), [3.3](#)
- [27] C. DESPRAT, H. LUGA et J.-P. JESSEL : Hybrid client-server and P2P network for web-based collaborative 3D design. *WSCG 2015 Conference on Computer Graphics, Visualization and Computer Vision*, p. 229–238, 2015. [2.2.4](#), [3.1](#), [3.2.1](#), [4.3.2](#), [5.2](#), [5.3](#)

- [28] D. DIAS : browserCloud.js - A federated community cloud served by a P2P overlay network on top of the web platform. (May), 2015. [2.2.4](#)
- [29] F. A. EKADIYANTO et A. HUNGER : 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, 2012. [2.2.5](#)
- [30] C. A. ELLIS et S. J. GIBBS : Concurrency control in groupware systems. *ACM SIGMOD Record*, 18(2):399–407, 1989. [1.1.3](#), [2.1.1](#)
- [31] E. EVANS : *Domain-Driven Design : Tackling Complexity in the Heart of Software*. Addison Wesley, 2003. [2.3.3](#), [2.3.3](#)
- [32] C. GADEA, D. HONG, D. IONESCU et B. IONESCU : 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)*, p. 1–6, 2016. [2.1.3](#)
- [33] M. GOTTA : Categorizing Collaboration, 2007. [1.1.1](#)
- [34] H. GRASBERGER, P. SHIRAZIAN, B. WYVILL et S. GREENBERG : 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*, p. 29, 2013. [2.1.2](#), [2.2](#)
- [35] S. GREENBERG et D. MARWOOD : Real time groupware as a distributed system. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work - CSCW '94*, p. 207–217, New York, New York, USA, 1994. ACM Press. [2.1.1](#)
- [36] I. J. GRIMSTEAD, D. W. WALKER et N. J. AVIS : Collaborative visualization : A review and taxonomy. *Proceedings - IEEE International Symposium on Distributed Simulation and Real-Time Applications, DS-RT*, p. 61–69, 2005. [2.1.2](#)
- [37] Y.-U. HA, J.-H. JIN et M.-J. LEE : Lets3D : A Collaborative 3D Editing Tool Based On Cloud Storage. *International Journal of Multimedia and Ubiquitous Engineering*, 10(9):189–198, 2015. [2.1.1](#)
- [38] C. HAND : A Survey of 3D Interaction Techniques. *Computer Graphics Forum*, 16(5):269–281, 1997. [1.1.2](#)
- [39] S. HELMER, A. POULOVASSILIS et F. XHAFA : *Reasoning in event-based distributed systems*. 2011. [2.3.1](#)
- [40] A. HINZE, K. SACHS et A. BUCHMANN : Event-based Applications and Enabling Technologies. *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, p. 1 :1–1 :15, 2009. [1.1.4](#), [3.2](#)
- [41] S. HOGG : What About Stream Control Transmission Protocol (SCTP) ? [2.2.3](#)
- [42] B. HOUSTON, R. CHEN, T. MCKENNA, W. LARSEN, B. LARSEN, J. CARON, N. NIKFETRAT, C. LEUNG, J. SILVER, H. KAMAL-AL-DEEN et P. CALLAGHAN : Clara.io. *ACM SIGGRAPH 2013 Studio Talks on - SIGGRAPH '13*, p. 1–1, 2013. [2.1.2](#), [2.2](#)

- [43] S. Y. HU, T. H. HUANG, S. C. CHANG, W. L. SUNG, J. R. JIANG et B. Y. CHEN : FLoD : A framework for peer-to-peer 3D streaming. *Proceedings - IEEE INFOCOM*, p. 2047–2055, 2008. [4.3.2](#)
- [44] J. JIA, W. WANG et X. HEI : An efficient caching algorithm for peer-to-peer 3D streaming in distributed virtual environments. *Journal of Network and Computer Applications*, 42:1–11, 2014. [4.3.2](#)
- [45] Y. JUNG, J. BEHR, T. DREVENSEK et S. WAGNER : Declarative 3D approaches for distributed web-based scientific visualization services. *CEUR Workshop Proceedings*, 869, 2012. [2.1.3](#)
- [46] KHRONOS : OpenGL ES 2. 2007. [2.1.3](#)
- [47] KHRONOS : OpenGL ES 3. 2008. [2.1.3](#)
- [48] KHRONOS : WebGL 1.0. 2011. [2.1.3](#)
- [49] KHRONOS : WebGL 2.0, 2016. [2.1.3](#)
- [50] J. KLAMER : *Conflict resolution in an event sourcing environment*. Thèse de doctorat, 2013. [2.3.5](#)
- [51] T. KOSKELA, A. HEIKKINEN, E. HARJULA, M. LEVANTO et M. YLIANTTILA : RADE : Resource-aware Distributed Browser-to- browser 3D Graphics Delivery in the Web. *IEEE Wireless and mobile*, p. 500–508, 2015. [3.2](#)
- [52] T. KOSKELA, J. VATJUS-ANTTILA et T. DAHL : Communication Architecture for a P2P-enhanced Virtual Environment Client in a Web Browser. p. 1–5, 2014. [2.2.4](#), [2.2.5](#)
- [53] Z. KOSMADOURI, T. LIM, J. RITCHIE, S. LOUCHART, Y. LIU et R. SUNG : Engineering design using game-enhanced CAD : The potential to augment the user experience with game elements. *CAD Computer Aided Design*, 45(3), 2013. [1.1.2](#)
- [54] S. KOUNEV, J. BACON, K. SACHS et A. BUCHMANN : 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*, p. 13–22, 2008. [2.3.2](#)
- [55] L. LAMPORT : Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(7):558–565, 1978. [3.2.3](#)
- [56] G. LAVOUÉ, L. CHEVALIER et F. DUPONT : Streaming compressed 3D data on the web using JavaScript and WebGL. *Proceedings of the 18th International Conference on 3D Web Technology - Web3D '13*, p. 19, 2013. [4.3.2](#)
- [57] J. LEWIS : IBM Computer Usability Satisfaction Questionnaires : Psychometric Evaluation and Instructions for Use. *International Journal of Human-Computer Interaction*, 7(1):57–78, 1995. [5.4](#)
- [58] J. R. LEWIS : Multipoint scales : Mean and median differences and observed significance levels. *International Journal of Human-Computer Interaction*, 5(4):383–392, 1993. [5.4](#)
- [59] J. LI, J.-K. CHOU et K.-L. MA : High performance heterogeneous computing for collaborative visual analysis. *SIGGRAPH Asia 2015 Visualization in High Performance Computing on - SA '15*, p. 1–4, 2015. [2.2.4](#), [2.3.2](#)

- [60] M. LIMPER, M. THÖNER, J. BEHR et D. W. FELLNER : SRC - a streamable format for generalized web-based 3D data transmission. *Proceedings of the Nineteenth International ACM Conference on 3D Web Technologies - Web3D '14*, p. 35–43, 2014. [4.3.2](#)
- [61] D. LOWET et D. GOERGEN : Co-Browsing Dynamic Web Pages. *Proceedings of the 18th International Conference on World Wide Web - WWW '09*, p. 941–950, 2009. [2.1.3](#)
- [62] Z. LU, P. GUERRERO, N. J. MITRA et A. STEED : Open3D : Crowd-Sourced Distributed Curation of City Models. *Web3D '16 : Proceedings of the 21th International Conference on 3D Web Technology*, p. 87–94, 2016. [2.1.2](#)
- [63] A. MAGLO, G. LAVOUE, F. DUPONT et C. HUDELOT : 3D mesh compression : survey , comparisons and emerging trends. *ACM Computing Surveys*, 9(4), 2013. [4.3.2](#)
- [64] A. MARTINEZ G., H. OROZCO, F. RAMOS et M. SILLER : 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*, 2009. [2.2.5](#)
- [65] E. MORIN : *Introduction à la pensée complexe*. ESF Editeur, 1990. [1](#)
- [66] E. MORIN : Science avec conscience, 1990. [1.1.1](#)
- [67] C. MOUTON, I. GRIMSTEAD et U. CARDIFF : Collaborative Visualization Current Systems and Future Trends. *Proceedings of the 16th International Conference on 3D Web Technology*, 1:101–110, 2011. [2.1.2](#)
- [68] C. MOUTON, S. PARFOURU, C. JEULIN, C. DUTERTRE, J.-L. GOBLET, T. PAVIOT, S. LAMOURI, M. LIMPER, C. STEIN, J. BEHR et Y. JUNG : Enhancing the Plant Layout Design Process using X3DOM and a Scalable Web3D Service Architecture, 2014. [2.2](#)
- [69] J. NIELSEN : *Usability Engineering*, vol. 1. AP Professional, 1993. [2.2](#)
- [70] M. OGDEN, K. MCKELVEY et M. B. MADSEN : Dat -Distributed Dataset Synchronization And Versioning. (May), 2017. [2.1.1](#)
- [71] B. OKI, M. PFLUEGL, A. SIEGEL et D. SKEEN : The Information Bus. *Proceedings of the fourteenth ACM symposium on Operating systems principles - SOSP '93*, p. 58–68, 1993. [2.3.2](#)
- [72] F. PACULL, A. SANDOZ et A. SCHIPER : Duplex : A Distributed Collaborative Editing Environment in Large Scale. *Proceedings of the 1994 ACM conference on Computer supported cooperative work - CSCW '94*, p. 165–173, 1994. [2.1.1](#)
- [73] N. PAPAGEORGIOU, Y. VERGINADIS, D. APOSTOLOU et G. MENTZAS : Collaboration pattern assistant. *Proceedings of the 5th ACM international conference on Distributed event-based system - DEBS '11*, p. 387, 2011. [2.3.1](#)
- [74] H. PARZYJEGLA, D. GRAFF, A. SCHRÖTER, J. RICHLING et G. MÜHL : 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)*, vol. 6462 LNCS, p. 124–140. Springer, Berlin, Heidelberg, 2010. [2.3.2](#)

- [75] P. PIETZUCH et J. BACON : Hermes : a distributed event-based middleware architecture. *22nd International Conference on Distributed Computing Systems Workshops (ICDCS 2002)*, p. 611–618, 2002. [2.3.2](#)
- [76] F. PONCHIO et M. DELLEPIANE : Fast decompression for web-based view-dependent 3D rendering. In *Web3D '15 : Proceedings of the 20th International Conference on 3D Web Technology*, p. 199–207, 2015. [4.3.2](#)
- [77] A. PRAKASH et M. J. KNISTER : A framework for undoing actions in collaborative systems. *ACM Transactions on Computer-Human Interaction*, 1(4):295–330, 1994. [2.1.1](#), [2.3.5](#)
- [78] D. ROBERTO, C. DIAS, R. S. DURELLI, J. REMO, F. BREGA, B. B. GNECCO, L. C. TREVELIN et M. D. P. GUIMARÃES : Data Network in Development of 3D Collaborative Virtual Environments : A Systematic Review. *LNCS*, 8579:769–785, 2014. [2.2.5](#)
- [79] M. SHAPIRO et N. PREGUIÇA : Designing a commutative replicated data type. *arXiv preprint arXiv :0710.1784*, 2007. [2.1.1](#)
- [80] S. SINGHAL et M. ZYDA : Networked Virtual Environments. (April):222–226, 1999. [1.1.2](#)
- [81] K. SONS, F. KLEIN, D. RUBINSTEIN, S. BYELOZYOROV et P. SLUSALLEK : XML3D – Interactive 3D Graphics for the Web. *Proceedings of the 15th International Conference on Web 3D Technology*, p. 175–184, 2010. [2.1.3](#)
- [82] M. STEIAKAKI, K. KONTAKIS et A. G. MALAMOS : Real-Time Collaborative environment for interior design based on Semantics , Web3D and WebRTC. *International Symposium on Ambient Intelligence and Embedded Systems*, p. 22–24, 2016. [2.2.4](#)
- [83] C. STEIN, M. LIMPER et A. KUIJPER : 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*, p. 53–61, 2014. [2.1.3](#)
- [84] C. STEINFELD, C.-Y. JANG et B. PFAFF : Supporting Virtual Team Collaboration - TeamSCOPE System. *International Conference on Supporting Group Work (GROUP'99)*, p. 81–90, 1999. [1.1.4](#)
- [85] C. SUN : Undo as concurrent inverse in group editors. *ACM Transactions on Computer-Human Interaction*, 9(4):309–361, 2002. [2.1.1](#), [2.3.5](#)
- [86] C. SUN, X. JIA, Y. ZHANG, Y. YANG et D. CHEN : Achieving Convergence , Causality Preservation , and Intention Preservation in Real-Time Cooperative Editing Systems. *ACM Trans. Comput.-Hum. Interact.*, 5(1):63–108, 1998. [1.1.3](#), [3.3](#)
- [87] C. SUN, Y. ZHANG, X. JIA et Y. YANG : A generic operation transformation scheme for consistency maintenance in real-time cooperative editing systems. *Proceedings of the international ACM . . .*, p. 425–434, 1997. [1.1.3](#)
- [88] M. Y. SUNG, Y. YOO, K. JUN, N. J. KIM et J. CHAE : Experiments for a collaborative haptic virtual reality. *Proceedings - 16th International Conference on Artificial Reality and Telexistence - Workshops, ICAT 2006*, p. 174–179, 2006. [2.2.5](#)

- [89] J. SUTTER : A CSS Integration Model for Declarative 3D. In *Web3D '15 : Proceedings of the 20th International Conference on 3D Web Technology*, p. 209–217, 2015. [2.1.3](#)
- [90] S. TARKOMA : Research Solutions. In *Publish/Subscribe Systems*, p. 205–237. John Wiley & Sons, Ltd, Chichester, UK, 6 2012. [2.3.2](#)
- [91] C. TOMINSKI : Event-Based Visualization for User-Centered Visual Analysis. *Computer*, 2006. [3.2.1](#)
- [92] Y. VERGINADIS, D. APOSTOLOU, N. PAPAGEORGIOU et G. MENTZAS : Collaboration Patterns in Event-Driven Environments for Virtual Organizations. Rap. tech., 2009. [2.3.1](#)
- [93] V. VERNON : *Implementing Domain-Driven Design*. Addison-Wesley Longman Publishing Co., Inc., 2013. [2.3.3](#)
- [94] N. VIDOT : *Convergence des copies dans les environnements collaboratifs répartis*. Thèse de doctorat, 2002. [1.1.3](#)
- [95] W3C : Extensible 3d (X3D), 2011. [2.1.3](#)
- [96] S. WEISS, P. URSO et P. MOLLI : An Undo Framework for P2P Collaborative Editing. In E. BERTINO et J. B. D. JOSHI, éds : *Collaborative Computing : Networking, Applications and Worksharing : 4th International Conference, CollaborateCom 2008, Orlando, FL, USA, November 13-16, 2008, Revised Selected Papers*, p. 529–544. 2009. [2.1.1](#), [2.3.5](#)
- [97] S. WEISS, P. URSO et P. MOLLI : Logoot-undo : Distributed collaborative editing system on P2P networks. *IEEE Transactions on Parallel and Distributed Systems*, 21(8):1162–1174, 2010. [1.1.3](#), [2.3.5](#)
- [98] D. WU, D. W. ROSEN et D. SCHAEFER : *Cloud-Based Design and Manufacturing (CBDM)*. 2014. [1.1](#)
- [99] F. XHAFA et A. POULOVASSILIS : 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, 2010. [1.1.4](#)
- [100] Y. YOU et S. PEKKOLA : Meeting others—supporting situation awareness on the WWW. *Decision Support Systems*, 32:71–82, 2001. [1.1.4](#)
- [101] G. YOUNG : Code Better, 2009. [2.3.4](#)
- [102] G. YOUNG : CQRS and CAP Theorem, 2010. [3.2.3](#)
- [103] M. YU, H. CAI, X. MA et L. JIANG : Symmetry-Based Conflict Detection and Resolution Method towards Web3D-based Collaborative Design. *Symmetry*, 8(5):35, 5 2016. ([document](#)), [2.1](#)
- [104] P.-p. Y. P.-p. YUAN, G. C. G. CHEN, J.-x. D. J.-x. DONG et W.-l. H. W.-l. HAN : 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, 2002. [2.3.1](#)
- [105] L. ZHANG, F. ZHOU, a. MISLOVE et R. SUNDARAM : Maygh : Building a CDN from client web browsers. *Proceedings of the 8th ACM European Conference on Computer Systems, EuroSys 2013*, p. 281–294, 2013. [2.2.4](#)

Bibliographie

- [106] M. ZHU, S. MONDET, G. MORIN, W. T. OOI et W. CHENG : Towards peer-assisted rendering in networked virtual environments. *Proceedings of the 19th ACM international conference on Multimedia - MM '11*, p. 183, 2011. [2.2.5](#)