

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

Caroline Desprat

25 septembre 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 réunit dans un même espace partagé des utilisateurs distants géographiquement pour un objectif de collaboration commun, les ressources qu'ils apportent (calcul, stockage ...) sont encore trop rarement utilisées et cela constitue un défi. Il s'agit en effet de proposer un système simple, efficace 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 hétérogène 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 de ces environnements collaboratifs 3D : la dynamique des utilisateurs et leur nombre, le type de données 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'événements. 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 événementiel, 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 ressources of their clients (calcul, storage ...) 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 environments 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 intrinsic 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 architectures orientées évènements pour la collaboration	7
1.2 Problématique	12
1.3 Contributions	14
1.3.1 Contributions théoriques	14
1.3.2 Contributions pratiques	15
1.4 Organisation du manuscrit	15
2 État de l'art	17
2.1 Modélisation 3D collaborative	18
2.1.1 Collaboration et concurrence dans un Environnement Virtuel 3D	18
2.1.2 ... sur le web : HTML5 et au delà	19
2.1.3 Web 3D : deux approches	22
2.2 Communication en temps-réel	25
2.2.1 Fiable versus Non Fiable	25
2.2.2 Ordonné versus Désordonné	27
2.2.3 Les principaux protocoles de transport	27
2.2.4 Quel protocole dans quel EVC3D ?	29
2.3 Les systèmes distribués orientés évènements pour la collaboration	32
2.3.1 Introduction	32
2.3.2 Systèmes Publish-Subscribe	32
2.3.3 Domain Driven Design	38
2.3.4 Command Query Responsibility Segregation	40
2.3.5 Event Sourcing	42
2.3.6 Bilan	46
2.4 Conclusion	47

3	Contributions scientifiques	49
3.1	Introduction	50
3.2	Modèle évènementiel pour l'intégration du domaine 3D dans les EVC	50
3.2.1	Modèle général	53
3.2.2	Mécanisme de gestion de version	56
3.2.3	Cohérence Éventuelle en CQRS	56
3.2.4	Potentielles applications et autres utilisations	57
3.2.5	Bilan	58
3.3	Architecture de communication hybride	58
3.3.1	Présentation générale	61
3.3.2	Event Store distribué	62
3.3.3	Persistance à long terme	64
3.3.4	Synchronisation client-serveur	64
3.3.5	Gestion de la cohérence	65
3.3.6	Bilan	65
3.4	Conclusion du chapitre	65
4	Implantation	67
4.1	Introduction	68
4.2	3DEvent : Plateforme web de manipulation collaborative d'objets 3D	68
4.2.1	Éditeur 3DEvent	68
4.2.2	Interface utilisateur	69
4.2.3	Flexibilité de la visualisation	70
4.3	Intergiciel P2P pour l'échange de données 3D	71
4.3.1	Données d'échange	72
4.3.2	Synchronisation des données	72
4.4	Résumé des choix techniques	74
4.5	Bilan	74
5	Expérimentations	75
5.1	Introduction	76
5.2	Cas d'étude : Assemblage collaboratif d'objets 3D dans un environne- ment web	76
5.3	Expérimentation 1 : preuve de faisabilité	76
5.3.1	Présentation de l'expérimentation	76
5.3.2	Résultats	76
5.3.3	Discussion et Conclusion	76
5.4	Expérimentation 2 : Intégration du framework évènementiel	76
5.4.1	Résultats et Discussion	79
5.5	Comparaison entre l'expérimentation 1 et l'expérimentation 2	81
5.5.1	Résultats	81
5.5.2	Discussion et Conclusion	81
5.6	Bilan	81
6	Conclusion	87

Bibliographie	89
----------------------	-----------

Table des figures

1.1	Place de la contributions dans les différents champs de recherche . . .	13
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	Architecture Publish-Subscribe	33
2.4	Illustration du patron Domain Driven Design (DDD) et de ses artefacts (issue de [?] sous licence Creative Commons)	39
2.5	Architecture en 4 couches du DDD (gauche) en miroir avec l'architec- ture CQRS (droite)	41
2.6	Transaction en Event-Sourcing	43
2.7	Transaction avec compensation en Event-Sourcing	43
2.8	Snapshot en Event-Sourcing	44
3.1	Exemple d'arbre de commits Git	58
3.2	Pile des protocoles IP: TCP vs UDP	60
3.3	Composants de chaque pair dans 3DEvent (point de vue réseau) . . .	62
3.4	Protocole de connexion au réseau d'instance 3DEvent	63
5.1	Interface utilisateur pendant une session collaborative (trois personnes)	82
5.2	Persistance long-terme (Event Store [®]), base de données/outil de mo- nitoring	83
5.3	Résumé d'une session collaborative au cours du temps	84
5.4	Résultats des questionnaires collectés	85

Liste des tableaux

2.1	Solutions conventionnelles pour les problèmes collaboratifs [Yu <i>et al.</i> , 2016].	19
2.2	Encodage des données transmises	24
2.3	Aperçu des protocoles de transport	29
4.1	Type de messages lors de la synchronisation	73
4.2	Statut du nœud	74
5.1	Modèles utilisés durant l'expérimentation	77

Liste des abréviations

3D tridimension. 2, 12–15, 59

AIC Architecture, Ingénierie et Construction. 18

API Application Programming Interface. 19, 22, 57, 69, 70

BIM Business Information Modeling. 3, 4, 6, 12, 18

CAO Conception Assistée par Ordinateur. 2, 6, 12, 15, 18–20, 38

CAP Consistency, Availability, and Partition Tolerance. 55

CCI Causality, Convergence, Intention. 57

CE cohérence éventuelle. 54

CEP Complex Event Processing. 56

cloud Définir le cloud. 50

CQRS Command Query Responsibility Segregation. 39, 40, 48–50, 55, 68

CS Command Sourcing. 44, 50

CSS Cascading Style Sheet. 21

DDD Domain Driven Design. v, 36–38, 48

DDS Data Distribution Service. 35, 36

DOM Document Object Model. 21, 22

EP event processing. 49

ES Event Sourcing. 38, 40–44, 48, 49, 51, 55, 56

EV Environnement Virtuel. 23, 28

EVC Environnement Virtuel Collaboratif. 23, 29, 59

EVC 3D Environnement Virtuel Collaboratif 3D. 2, 4–7, 10, 14, 15, 23, 24, 28, 29, 48, 57, 63

event store Mémoire permettant le stockage des évènements en mode "*append-only*". 54

Framework (ou structure logicielle en français) est un ensemble de composants génériques proposant un cadre de travail guidant l'architecture logicielle. 14, 30, 54, 55, 66, 67

- glTF** GL Transmission Format. 70
- HTML** HyperText Markup Langage. 21, 22, 68
- ICE** Interactive Connectivity Establishment. 35
- IETF** Internet Engineering Task Force. 57
- IU** Interface Utilisateur. 33, 34, 67
- JSON** JavaScript Object Notation. 70
- LAN** Local Area Network. 25, 28, 29
- MAN** Metropolitan Area Network. 25
- NAT** Network Address Translator. 35
- NAT Traversal** est une technique pour établir et maintenir les connexions internet à travers les passerelles qui implémentent [Network Address Translator \(NAT\)](#). [NAT](#) casse le principe de connectivité de bout en bout originalement envisagée lors de la conception d'internet.. 35
- NoSQL** Not Only SQL. 72
- OWL** Web Ontology Language. 30
- P2P** pair à pair. 8, 9, 15, 19, 29–35, 50, 56–60, 68
- PDM** Product Data Management. 18
- PLM** Product Lifecycle Management. 6, 18, 20
- PubSub** Publish-Subscribe. 10, 31–33, 36, 44
- RTCP** Real-time Transfert Control Protocol. 23, 24, 29
- RTP** Real-Time Transport Protocol. 23, 24, 29
- RV** Réalité Virtuelle. 18
- SCTP** Stream Control Transmission Protocol. 27–29
- SEC** Système d'Édition Collaborative. 26, 56, 57
- 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. 71
- SVG** Scalable Vector Graphics. 21
- TCP** Transmission Control Protocol. 26–29, 57
- UDP** User Datagram Protocol. 26–29, 57

W3C World Wide Web Consortium. [15](#), [34](#), [57](#)

WAN Wide Area Network. [25](#), [28](#)

WebRTC Web Real-Time Communication. [33](#), [34](#), [50](#), [57](#), [69](#)

WebSocket Protocole réseau standard du Web visant à créer des canaux de communication full-duplex par dessus une connexion TCP. [19](#), [57](#), [69](#)

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 architectures orientées évènements pour la collaboration	7
1.2	Problématique	12
1.3	Contributions	14
1.3.1	Contributions théoriques	14
1.3.2	Contributions pratiques	15
1.4	Organisation du manuscrit	15

1.1 Contexte

Le besoin de collaboration et de manipulation en temps-réel d'objets en **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 [Chandrasegaran *et al.*, 2013, Wu *et al.*, 2014]. Or, la quantité et la taille des données étant toujours croissante, il devient de plus en plus difficile de partager les modèles, particulièrement avec les utilisateurs n'ayant pas accès aux derniers logiciels et matériels graphiques. En conséquences, le développement de plateformes légères déployées sur le web pour répondre à ces besoins est en forte progression. Dans un **Environnement Virtuel Collaboratif 3D (EVC 3D)**, la simulation distribuée d'un environnement virtuel 3D, un grand nombre d'utilisateurs situés à différents endroits géographiques peuvent interagir les uns avec les autres en temps-réel. Cette distorsion de l'espace physique et temporel impose aux **EVC 3D** des mécanismes de communication rapide, 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 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 de taille 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 deux (ou plus) personnes (ou organisations) travaillent ensemble à la croisée de buts communs en partageant leurs connaissances pour apprendre et bâtir un consensus. La collaboration permet l'émergence de conceptions partagées dans la réalisation de visions partagées dans des environnements et des systèmes complexes. Les imbrications de

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

chaque domaine et la transdisciplinarité sont acceptés comme dans le concept de pensée complexe. D'ailleurs, dans sa définition de la complexité, Edgar Morin fait référence au sens étymologique latin « *complexus* » qui signifie « ce qui est tissé ensemble » [Morin, 1990b]. La plupart des collaborations requièrent un élément dirigeant qui peut prendre une forme sociale (personne) au sein d'un groupe décentralisé et égalitaire (tout le monde au même niveau). L'élément dirigeant va souvent aider également trouver des consensus. La disponibilité des ressources peut également devenir un élément dirigeant dans la collaboration. Une équipe travaillant de manière collaborative peut 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 aurait été incapables (temps, espace, connaissance, matériel) d'accomplir le travail seul.

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

- *Collaboration centrée processus*. Les conditions requises du processus nécessitent l'engagement de l'utilisateur, qui doit, de part 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 communiquer à propos.
- *Collaboration centrée activité*. Les activités partagées créent un sentiment de co-dépendance qui motive la collaboration entre les membres. La co-dépendance

prend l'avantage sur le propre intérêt de chacun comme motivation pour 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'ouragan ou de tsunami), 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 avec différentes compétences / connaissances qui doivent s'entraider pour parvenir à la mise en commun des objets 3D et réaliser leur objectif.

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

- *Collaboration par équipe.* Dans une équipe tous les membres se connaissent. Il y a une interdépendance claire des tâches à effectuer où la réciprocité est attendue, avec un échéancier et des objectifs explicites. Pour réaliser son but,

l'équipe doit réaliser les tâches dans un temps imparti. La collaboration par équipe suggère que les membres coopèrent sur un pied d'égalité (bien qu'il y ait souvent un chef) recevant une reconnaissance égale.

- *Collaboration communautaire.* L'objectif de ce type de collaboration est plus orienté sur la possibilité d'apprendre que sur la tâche elle-même, même si les centres d'intérêt sont partagés par la communauté. Les utilisateurs sont là pour partager et construire la connaissance plus que compléter un projet. Les membres vont aller voir leur communauté pour demander de l'aide sur un problème ou un avis et ramener 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 peuvent avoir des statuts privilégiés. La réciprocité est un facteur important dans la communauté pour que cela fonctionne.
- *Collaboration en réseau.* La collaboration en réseau est une sur-couche 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, des accès à internet omniprésents et la capacité de se connecter avec divers individus malgré la distance.

Cette thèse, en se référant à cette seconde classification, s'intéresse plutôt à 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'environnement 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 plateforme comme le partage d'espace, de présence et du temps. Selon Singhal et Zyda [Singhal et Zyda, 1999], **EVC 3D** est constitué de quatre composants principaux : (i) un moteur graphique pour l'affichage ; (ii) des appareils de contrôle et de communication ; (iii) un système de traitement ; (iv) et un réseau de transmission des données.

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

La manipulation d'objet est 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 au développement de ces activités. La manipulation collaborative s'avère indispensable dans plusieurs types d'applications comme le prototypage virtuel, les simulations d'entraînement ou la simulation d'assemblage et de maintenance. L'expérience de modélisation **CAO** collaborative peut être accompagnée de mécanismes ludiques pour intégrer une capture temps-réel de la connaissance [Kosmadoudi *et al.*, 2013]. Ces lieux virtuels sont l'occasion pour les utilisateurs de participer de manière naturelle et efficace à la création et à la vie de l'objet manipulé dans l'environnement virtuel sans danger et à bas coût. Une autre utilisation des **EVC 3D** sert la navigation virtuelle, c'est à dire les visites collaboratives (musées,

héritage culturel, les revues de projet architecturaux / urbains, ou encore les jeux collaboratifs (cours, simulateur de jeux collectifs). Les [EVC 3D](#) permettent non seulement aux utilisateurs de communiquer de manière distante mais également d'interagir ensemble en partageant 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 les protocoles réseaux) et d'IHM doivent être résolus pour concevoir un [EVC 3D](#) fiable, ergonomique et en temps-réel.

1.1.3 Les architectures orientées événements pour la collaboration

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

Dans la littérature, une variété de définitions du terme événement existe. Habituellement, un événement est considéré comme quelque chose qui « arrive », en particulier quelque chose d'inhabituel ou d'important. Cependant la plupart des travaux de la littérature évitent de définir le sens précis des événements en n'indiquant pas le champ d'application dans lequel ils sont utilisés. Il existe trois entrées pour le terme événement dans le dictionnaire. La première entrée se réfère à la physique (dans le cadre de la théorie de la relativité) où un événement est « un phénomène considéré comme localisé et instantané, survenant en un point et un instant bien déterminés ». La seconde se réfère à la théorie des probabilités, indiquant qu'un événement est « la partie d'un univers Ω réalisée quand l'une des éventualités la composant se réalise ». La troisième définition se rapporte à la psychologie impliquant « tout ce qui est capable de modifier la réalité interne d'un sujet (fait extérieur, représentation, etc.) ». Cette dernière définition est très proche de ce qu'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 comprends ses propres associations au terme évènement.

L'implantation d'un système orienté évènements 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 basés évènement doivent répondre aux exigences des utilisateurs et aux problèmes liés à la nature des plateformes et application. 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 indépendant de la localisation des services. Le développement d'application dirigées par les évènements est un challenge tripartite : la production d'évènement, le traitement des évènements et la consommation des évènements [Chandy *et al.*, 2011]. Cristea *et al.* [Cristea *et al.*, 2011] présentent un aperçu des architectures distribuées pour les systèmes basés 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 des modèles basés évènements prenant en compte la sensibilisation au groupe (*awareness*) dans des activités. Ces systèmes sont présents dans la littérature depuis les prémices de la technologie web [Bentley *et Wakefield*, 1997, Steinfield *et al.*, 1999, You *et Pekkola*, 2001]. Ces propositions 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

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

ubiquitaire 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 groupe 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 [Xhafa et Poulovassilis, 2010] exposent une approche distribuée basée événements pour gérer des collecticiels (*groupware*) en P2P. Leur méthode permet de développer des applications de collecticiels spécifiques sur les opérations et les services primitifs liés à la sensibilisation. Ces derniers sont directement implémentés dans l'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 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 dynamique des événements Les systèmes P2P sont dynamique par nature (*join / leave* des pairs). La fonctionnalité de sensibilisation doit prendre en compte cette dynamique. 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 dynamique du réseau doit intégrer des mécanismes de propagation et de réplication fournissant le contenu au groupe.

La généricité des événements Un système basé événements manipule plusieurs types d'événement. C'est pourquoi il est nécessaire que les événements soient le plus génériques possible 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 causé 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.

Dans [Brown *et al.*, 2003], les auteurs 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 réplique 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 [Hinze *et al.*, 2009]. 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 du fait de l'usage de données confidentielles ou critiques.

Intégration des contraintes métiers

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.

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 aux attentes utilisateurs (rétro-compatibilité, temps de chargement), elle varie selon le domaine mais reste

très assez prégnante sur le web en général. Par exemple, dans un domaine annexe comme le e-commerce, une étude réalisée en 2009 explique qu’une grande partie des internautes abandonnaient leurs achats en ligne si les pages mettaient 2 secondes ou plus à charger³. De nos jours, ce délai a été réduit au quart de seconde pour les grandes entreprises du web. Jakob Nielsen [?] indique trois temps de réponse 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.⁴
- *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 ; l’utilisateur 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 de l’application web et des données relatives mais également d’affichage des modèles 3D sont spécifiques dans ce domaine car ils ont tendance à être lourds (structure de données 3D et métadonnées) à charger (contrairement au texte par exemple). Cette phase peut s’accorder des 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 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 –

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

4. 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.

produites par les collaborateurs – les latences réseaux rentrent en comptes (serveur, pairs...) on s'accordera sur un délai entre 1 et 10 secondes dans cette thèse selon les degrés d'asynchronicité possibles.

Les bureaux d'études en ingénierie et en architecture travaillent sur des projets (visualisation [CAO](#), [BIM](#), gestion et arrangement d'espaces architecturaux) qui nécessitent la collaboration de professionnels venant de milieux différents avec des compétences et connaissances variées. Les modifications dans leurs 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. Toutes ces entités ont besoin d'être capables de charger les ressources pour pouvoir les inspecter et les analyser. Ces contraintes se retrouvent dans le domaine du [BIM](#), l'architecture, l'héritage culturel, ou plus généralement dans des milieux transdisciplinaires concentrés sur la [3D](#). L'évolution de ces ressources passent par une visualisation et une manipulation collaborative efficace en terme de chargement de ressources et de transmission des mises à jour.

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 derniers 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. Le 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 dispositions, tels 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 client et procurant également plus d'autonomie aux utilisateurs utilisant leurs ressources propres. L'architecture de communication et les contraintes liées aux architectures distribuées et décentralisée correspond au second axe de recherche. Enfin, le dernier champs s'adresse à la partie que nous appellerons « métier » qui se rapporte au domaine de la manipulation d'objets 3D qui inclue la gestion du cycle de vie des données 3D de l'interaction utilisateur à son stockage en passant par sa distribution. Ces aspects se rapportent principalement à l'historique, la traçabilité de l'information et l'expertise embarquée dans le système dans le cadre de cette thèse.

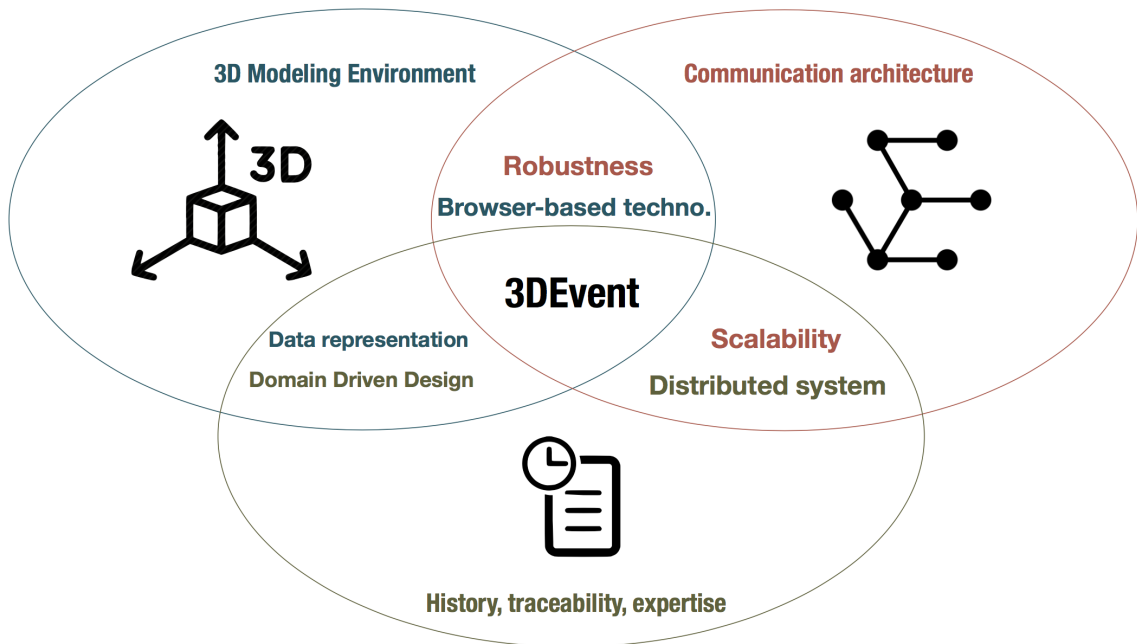


FIGURE 1.1 – Place de la contributions dans les différents champs de recherche

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

pour l'utilisateur, (iii) proposer un [framework](#) pour un [EVC 3D](#) web intégrant ces contraintes réseau, métier, et [3D](#) dans un navigateur. Dans le but d'achever de tels 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 à la fois 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. Pour ce faire, 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-réelle.
- 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 3D Event Client et de son évaluation utilisateur
- Implémentation d’un prototype 3D Event Architecture et de son évaluation

1.4 Organisation du manuscrit

La présence de la 3D et des environnements virtuels collaboratifs 3D est de plus en plus présents dans l’industrie de la CAO nécessitant de nouveaux modèles, outils et méthodes facilitant la mobilité et l’autonomie des utilisateurs. Il est aussi important de noter que le contenu multimédia 3D est aussi de plus en plus présent dans notre quotidien. Pour tout cela il est important de développer de nouveaux outils pour créer manipuler et partager du contenu 3D. 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 à besoin 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 3D Event pour manipuler des objets 3D avec une grande traçabilité et de manière autonome. Finalement, le chapitre adresse le problème et la spécificité de la transmission du contenu 3D lors de session collaborative en temps-réel. En considérant et développant un modèle orienté évènements, 3D Event dérive des stratégies pour délivrer et obtenir une scène 3D en continu (*streaming* en anglais) en s’appuyant sur les différentes configurations matérielles, utilisateurs, réseaux.

Chapitre 2

État de l’art

Contents

2.1	Modélisation 3D collaborative	18
2.1.1	Collaboration et concurrence dans un Environnement Virtuel 3D	18
2.1.2	... sur le web : HTML5 et au delà	19
2.1.3	Web 3D : deux approches	22
2.2	Communication en temps-réel	25
2.2.1	Fiable versus Non Fiable	25
2.2.2	Ordonné versus Désordonné	27
2.2.3	Les principaux protocoles de transport	27
2.2.4	Quel protocole dans quel EVC3D ?	29
2.3	Les systèmes distribués orientés événements pour la collaboration	32
2.3.1	Introduction	32
2.3.2	Systèmes Publish-Subscribe	32
2.3.3	Domain Driven Design	38
2.3.4	Command Query Responsibility Segregation	40
2.3.5	Event Sourcing	42
2.3.6	Bilan	46
2.4	Conclusion	47

2.1 Modélisation 3D collaborative

2.1.1 Collaboration et concurrence dans un Environnement Virtuel 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 coût 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 souhaité (forte, éventuelle) par exemple.

Les différentes solutions pour la conception collaborative peuvent être divisés en trois types : (i) les mécanismes de type autoritaires (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 comptés 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 [[Ha et al., 2015](#)] est un éditeur collaboratif 3D sur le web dont la gestion de la concurrence repose sur la demande de

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

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	Détection de conflit ou résolution avec interaction utilisateur
	Signalisation des conflit	
	Résolution de conflit créative	

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 [Ellis et Gibbs, 1989, Greenberg et Marwood, 1994]. Alors que Duplex [Pacull *et al.*, 1994], un [EVC 3D](#), est présenté au même moment, se 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 environnements distribué collaboratif [Prakash et Knister, 1994, Sun, 2002, ?]. Les algorithmes souvent utilisés dans ce cadre sont les algorithmes de transformées opérationnelles – [OT](#) – [Ellis et Gibbs, 1989] ou les algorithmes de réplication de données commutatives – [Commutative Replicated Data Types \(CRDTs\)](#) – [Shapiro et Preguiça, 2007] s'orientant de plus en plus vers des solutions décentralisées pour un usage à grande échelle [Weiss *et al.*, 2009].

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** à l'avantage d'être distribuée pour être accessible depuis un navigateur web ; avec un système d'édition hors-ligne, le téléversement peut s'avérer long et fastidieux. De plus, les logiciels spécialisés pour la modélisation 3D, contrairement aux jeux multijoueurs, sont traditionnellement dédiés à un utilisateur unique sans représentation visuelle des autres opérateurs dans le même espace 3D. Ici encore, les principes d'accessibilité du web facilitent la création d'un espace virtuel 3D collaboratif pour la visualisation, la manipulation et l'échange de données partagées.

La collaboration en temps-réel est de plus en plus présente sur le web pour différents types de documents notamment 3D. La revue sur la visualisation distribuée, effectuée par Grimstead et al. en 2005 [Grimstead *et al.*, 2005], 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. [Mouton *et al.*, 2011] 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 compre-

nant 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 autre que celle 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 :

1. développement : connaissance du standard [WebSocket](#) par les développeurs (historique)
2. gestion des données : simplification de la synchronisation des données (centralisé)
3. suivi de l’utilisateur : dépendance de l’utilisateur vis à vis de la connexion au serveur et donc du service (engagement)

Parmi les solutions commerciales comme OnShape, Clara.io et TinkerCAD le choix de plateformes web dans l’infonuagique pour la modélisation [CAO](#) est prépondérant souvent assorti d’un gestionnaire de version 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 modélisateurs autonomes (*standalone*) professionnels, le tout de manière collaborative. Leur approche de la gestion de version développe le concept de micro version [Baran, 2015]. Cette approche est employée dans les travaux de Lu et al. [Lu et al., 2016] pour gérer l’évolution de productions participatives liées à des tâches de modélisation 3D en usant de la créativité, de l’intelligence et du savoir-faire d’un grand nombre de personnes en sous-traitance (*crowdsourcing*). Le modélisateur 3D Clara.io [Houston et al., 2013] est plus généraliste. Il s’oriente davantage vers des fonctionnalités avancées de rendu 3D sur le *cloud* en utilisant du lancer de rayons. Les fonctionnalités d’historisation

et de collaboration proposées dans Clara.io restent très basiques (seulement annuler / refaire). TinkerCAD est aussi une plateforme de publication d'objets 3D pour faciliter l'accès à la conception d'objets 3D. TinkerCAD est une application destinée au grand public pour la conception et l'impression 3D. De ce fait, les fonctionnalités sont limitées à cause du 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. Un peu à part, GrabCAD est une plateforme web plus orientée sur la gestion de données 3D CAO (PLM) qui possède un système de gestion de version performant et un système d'annotation et de visualisation collaborative. 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) [Grasberger *et al.*, 2013]. Les objets sont alors manipulés comme des géométries de construction de solides avec les opérations booléennes associées. Le Tableau 2.2 résume les différents types d'encodage associés au type de modélisation 3D.

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 HyperText Markup Language (HTML)5 alors qu'en 3D, les spécifications sont encore en évolution.

L'approche déclarative (Dec3D) s'intègre au Document Object Model (DOM) et se focalise sur l'utilisation des technologies du web existantes comme CSS3, HTML5 et l'Ajax. X3D est ainsi un format de fichier respectant le standard ISO [W3C, 2011] pour représenter des scènes 3D interactives en XML, rétro-compatible avec VRML97. Il se différencie des formats de fichier type Collada en intégrant le comportement de la scène durant l'exécution en plus de la description du contenu 3D. Les deux bibliothèques les plus notables dérivées de ce standard sont X3DOM [Behr *et al.*, 2010] et XML3D



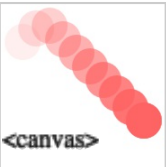

	2D (final HTML5 spec)	3D (No final W3C spec)
Déclaratif Graphe de scène Partie du HTML-document Intégration DOM CSS/ Evènements		
Impératif API procédurale Contexte de dessin		

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

[Sons *et al.*, 2010] : toutes les deux sont capables de supporter les récentes avancées de WebGL pour afficher une scène décrite dans le **DOM** à l'intérieur d'un *canvas* HTML5. X3DOM essaye de respecter le standard X3D et ses concepts pour en permettre l'intégration du format dans le **DOM**. De plus, X3DOM intègre le support d'élément **HTML**, les événements **DOM** et de profils **CSS** en supplément [Sutter, 2015]. En comparaison avec X3DOM, XML3D a développé une extension à **HTML5** pour décrire une scène 3D. L'utilisation d'un langage déclaratif basé sur XML comme X3D est bien adapté pour le contenu 3D utilise des données CAO/MAO ou WCS dans des applications **Système d'Information Géographique (SIG)** car il peut être directement transformé (ex. XSLT) d'une représentation à une autre (tout comme le X3D peut être rendu dans un navigateur en utilisant X3DOM).

L'approche déclarative est utilisée dans de nombreux travaux de visualisation scientifique 3D distribuée [Jung *et al.*, 2012] pour des données spatiales [Stein *et al.*, 2014] ou du rendu volumique [Becher, 2012] 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 [Gadea *et al.*, 2016] ou les événements du **DOM** [Lowet et Goergen, 2009] comme protocole de synchronisation de scènes. L'ajout de nombreux *listeners* sur un document peut affecter les performances de parcours de l'arbre du **DOM** notamment si une scène est très peuplée.

Concernant l'approche impérative, la spécification WebGL 1.0 [Khronos, 2011] proposée par le groupe Khronos permet aux navigateurs d'effectuer un rendu 3D grâce à une **API** JavaScript adaptée de l'**API** d'OpenGL ES 2.0 [Khronos, 2007] en utilisant l'élément *canvas* d'**HTML5**. Cette spécification est supportée par la



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

plupart des navigateurs traditionnels (Chrome depuis v49, Firefox depuis v52, Safari depuis v9.1, IE 11¹, Edge 14¹ et Opera depuis v41) ainsi que les navigateurs mobiles (iOS Safari depuis v9.3, Android Browser depuis v53 et Chrome for Android depuis v53) à l'exception d'Opera Mini². La spécification WebGL 2.0 [Khronos, 2016] basée sur OpenGL ES 3.0 [Khronos, 2008], déjà publiée en tant que brouillon, est en phase expérimentale³. 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 web⁴.

Tableau 2.2 – Encodage des données transmises

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

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

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

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

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

2.2 Communication en temps-réel

L'expression temps-réel est largement utilisé 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 [Environnement Virtuel Collaboratif \(EVC\)](#) particulièrement pour permettre aux utilisateurs de s'immerger dans la collaboration et faire confiance à l'application (exactitude des modifications).

La communication et la transmission des données lors de la collaboration un des aspects prépondérant dans les [EVC 3D](#) concerne . Que ce soit dans un [Environnement Virtuel \(EV\)](#), 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.

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-réelles. [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.

Du fait de la variété des domaines et applications auxquels s'adressent les [EVCs 3D](#) existants, il n'existe pas de protocole de communication parfait adapté à tous les types d'environnement. 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 à 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 ses données. 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 dépendant du type d'application et du type de l'environnement réseau dans lequel il est conçu pour opérer. Si une application dite temps-réel est conçue pour fonctionner en réseau local – [Local Area Network \(LAN\)](#) – l'utilisation d'un protocole fiable est une option recommandée. Les câbles utilisés dans ce contexte sont souvent rapides et stables; un paquet perdu ou corrompu peut rapidement être détecté et retransmis. Dans des environnements réseaux sont 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'ordonnement 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'ordonnement 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 commutatif.

2.2.3 Les principaux protocoles de transport

Les principes de fiabilité et d'ordonnement décrits plus haut sont souvent utilisés pour discriminer les différents protocoles de communication. Pour les applications en temps-réel, ils 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 leurs propriétés de fiabilité et d'ordonnement.

TCP

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

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

UDP

Le **User Datagram Protocol (UDP)** est considéré comme l'opposé de **TCP** concernant plusieurs aspects. Premièrement, **UDP** ne nécessite pas, pour établir la connexion, 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 établi par les deux terminaux, chacun peut envoyer ou recevoir des données de n'importe quel socket proprement configuré. Deuxièmement, **UDP** est un protocole dit « orienté message ». Par rapport à **TCP** où il n'y a pas de distinction claire entre les messages du fait de sa nature (flux continu), **UDP** fait une distinction claire entre chaque 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. . .).

STCP

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. La fiabilité peut aussi être configurée par l'utilisateur (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[?].

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)
STCP	Configurable	Configurable	Oui

2.2.4 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 [Roberto *et al.*, 2014], le protocole le plus souvent rencontré dans les **EVC 3D** est l'**UDP**. Cette prépondérance s'explique d'après deux situations. Premièrement, dans des réseaux où l'accès est fiable car ces environnements n'ont pas (ou peu) de perte de paquet (ex : **LAN**). Deuxièmement, dans un contexte d'application où la perte de paquet n'est pas critique (ex : jeu vidéo, diffusion en continu de son, vidéo, 3D...). **UDP** peut également s'utiliser dans les tout type de réseaux – **LAN** ou **WAN** (comme internet) –, car il est basé sur des datagrammes. Par ailleurs, le fait qu'il n'effectue pas de vérification de délivrance lui donne un avantage quant à la taille des données envoyées dans les **EVCs**. La responsabilité de vérifier quels sont les paquets qui n'ont pas été délivrés et leur ordre d'arrivée incombe à l'application. **TCP** est le protocole qui vient en seconde position. Les **EVC 3D** qui sont implémentés sur des réseaux haut-débit ne sont pas affectés par les étapes de vérifications que nécessitent une connexion **TCP**. D'après [Sung *et al.*, 2006], l'utilisation du protocole **TCP** sur internet dans le cadre des **EVC 3D** n'est pas recommandée à cause de la taille de l'en-tête et du ACK qui réduisent le débit des paquets. Dans ce contexte, le protocole **UDP** obtient de meilleurs résultats dans le cas où le flux de données est local (**LAN**). Les protocoles les plus adaptés pour communiquer dans un **EVC 3D** sont cependant **SCTP** et **RTP/RTCP**. Ils combinent les fonctionnalités de **TCP** et **UDP** et sont optimisés pour les applications multimédia (flux et temps-réel). **SCTP** offre la possibilité de choisir entre un système de livraison fiable (pour les paquets clés par exemple) ou non fiable (pour des mises à jour normales).

Depuis plusieurs décennies, les architectures **P2P** sont utilisées dans les **EVC** 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. [Martinez G. *et al.*, 2009], utilisent la localisation d'un utilisateur pour transmettre uniquement les mises à jours à son plus proche voisin. Ce système d'assistance par les

pairs peut également être utilisé pour améliorer le rendu dans les environnements virtuels en ajustant la qualité du rendu en fonction du coût de calcul [Zhu *et al.*, 2011]. Koskela et al. [Koskela *et al.*, 2014] sont allés plus loin dans cette direction en proposant la méthode RADE (*Ressource-Aware P2P-assisted 3D Delivery Method*). RADE repose sur l'utilisation du P2P pour alléger le chargement et donc réduire le cours 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.

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.

Autres Dat [Ogden *et al.*, 2017] 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 agnostique⁵ dont la conception repose sur la garantie des principes suivant : 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 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.

5. 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.

2.3 Les systèmes distribués orientés événements pour la collaboration

2.3.1 Introduction

La plupart des systèmes basés événements sont très populaires lorsqu'il s'agit de collaboration. On retrouve par exemple la spécification d'événements composites pour le support de la collaboration dans la conception logicielle [Yuan *et al.*, 2002] ou encore la définition de patrons de conception dédiés à la collaboration dans les architectures basées événement [Verginadis *et al.*, 2009]. Dans ce dernier domaine Papageorgiou et al. [Papageorgiou *et al.*, 2011] proposent un assistant 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 [Bang *et al.*, 2010]. L'utilisation d'une architecture basée événement permet à l'application d'être très extensible car très peu couplée. Chaque instance CoDesign intègre un intergiciel appelé CoWare en charge de la synchronisation de contenus édités de manière concurrentes sur les différentes instances CoDesign ainsi qu'un module chargé de notifier les architectes de situations de modélisation conflictuelles. Les conflits sont catégorisés de trois manières différentes.

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 plateforme⁶.

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

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

événements qui l'intéressent. La Figure 2.3 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*.

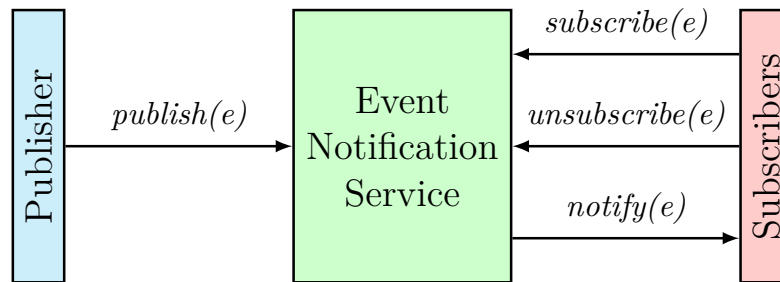


FIGURE 2.3 – Architecture Publish-Subscribe

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. Hermes [Pietzuch et Bacon, 2002] est un intergiciel (*middleware*) distribué basé événements. 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 événements, ce modèle rend le routage des données intuitif pour le développement d'applications distribuées à grande échelle comme le commerce en ligne (*e-commerce*). Dans le cadre de la gestion de groupe, Scribe [Castro *et al.*, 2002] propose un système **PubSub** pour gérer des groupes de souscripteurs et une architecture de communication *multicast* nécessaire à grande échelle pour envoyer les événements à tous les souscripteurs.

En juin 2017, Leonardo Quernozy propose 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 [Birman et Joseph, 1987] lorsqu'ils évoquent « the "News" service in the ISIS system » :

« *This service allows processes to enroll in a system-wide news facility. Each subscriber receives a copy of any messages having a "subject" for which it has enrolled in the order they were posted. Although modeled after net-news, the news service is an active entity that informs processes*

immediately on learning of an event about which they have expressed interest. » [Birman et Joseph, 1987]

Après cette publication les premiers systèmes [PubSub](#) commencent à se développer. Le concept de routeur d'information est introduite par Oki et al. [Oki *et al.*, 1993] : un seul bus d'information pour tous les services, objets, données ... Les protocoles de communication ont une sémantique minimale, les objets (instances de classe) sont auto-descriptifs, i.e. ils sont 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 [Banavar *et al.*, 1999], SIENA [Carzaniga *et al.*, 2000], REBECA [Parzyjeglą *et al.*, 2010] et également REDS, JEDI, PADRES présentés plus en détails dans [Tarkoma, 2012]. Ces différentes solutions ont été conçues comme des systèmes de gestion avec des gestionnaires d'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 dynamique.

Outils de monitoring et de benchmark

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 montée en charge. Carzaniga and Wolf [Carzaniga et Wolf, 2002] 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. [[Kounev et al., 2008](#)] ont utilisé des techniques d'analyse opérationnelle pour caractériser le trafic du système et dériver une approximation de la moyenne des délais de livraison d'évènements.

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

La virtualisation des clients

Une des problématiques soulevée par la collaboration [P2P](#) est de permettre la reproductibilité des expérimentations dans un environnement contrôlé et réaliste. Réussir à simuler un réseau virtuel de clients en [P2P](#) en utilisant le protocole [Web Real-Time Communication \(WebRTC\)](#) est un défi encore compliqué. Il existe des outils pour simuler des réseaux [P2P](#) tels que PeerSim [[Montresor et Jelasity, 2009](#)] ou *ns-3* [[Riley et Henderson, 2010](#)]. Ces outils sont plus orientés sur la façon de distribuer les informations lors de la simulation et leur dissémination au sein du réseau plutôt que de reproduire les protocoles et l'infrastructure de manière réaliste

mettre
ailleurs ?

avec des données issues de l'IU. Dans de récents travaux utilisant WebRTC, les expérimentations ne sont pas encore facilement reproductibles du fait qu'il n'existe pas d'outil facile à prendre en main pour effectuer ce genre de simulation à base d'entrées utilisateur fiables vis à vis des impératifs métier.

La virtualisation implique également de pouvoir simuler des comportements sur la base d'interactions issues de IU [Hu et Chen, 2017] comme on peut le trouver dans la simulation d'IU web. Dans ce contexte, ce type de tests permet de vérifier la compatibilité et la réactivité des différentes plateformes, versions de navigateurs et types d'appareils en fonction d'entrées utilisateur. C'est également utile pour faire des tests de performance ou de montée en charge concernant l'interface. Le service testRTC⁷ est un service payant qui propose un outil de test et de monitoring pour un grand banc de machines de sessions audio et vidéo WebRTC .

L'intérêt d'utiliser un modèle de réseau P2P virtuel comporte plusieurs avantages. En reprenant les points proposés par [Haque et al., 2016], on peut citer :

- Pas d'installation nécessaire. Plusieurs outils et logiciels existent pour simuler des réseaux P2P [Montresor et Jelasity, 2009] ou nécessite encore l'installation de clients lourds (clients BitTorrent) par les utilisateurs pour réaliser les mesures. Cela implique le fait de comprendre les principes de base concernant la configuration réseau (routeurs, pare-feu) et le protocole utilisé (BitTorrent). Très peu de travaux concernant WebRTC ont réussi à virtualiser les clients participants aux expérimentations.
- Opérabilité et interopérabilité dans un environnement contrôlé. L'installation d'un client sur une machine requiert certaines autorisations liés à la politique de l'organisation, la licence logicielle et le support logiciel. Ce type d'environnement est assez typique dans l'industrie, c'est pourquoi il est intéressant de proposer un modèle qui puisse s'exécuter sans difficulté grâce à l'utilisation de clients web. Les navigateurs qui servent de clients web s'accordent généralement avec les standards proposés par le W3C, ce qui facilite également l'interopérabilité du logiciel souvent déployé dans un parc hétérogène de machines.
- Indépendance de la situation géographique. Tout comme les infrastructures *cloud* (souvent un service tiers) qui sont distantes par l'intermédiaire d'un réseau , généralement internet les utilisateurs peuvent se connecter sur un réseau virtuel P2P à partir de n'importe quel lieu.

7. testrtc.com. Consulté le 07/07/2017

- Simplification de la maintenance. Les applications, standards et protocoles autour du [P2P](#) sont en constante évolution. L'implémentation de la méthode de distribution des données nécessite par conséquent de fréquentes mises à jour pour être la plus efficace possible. Dans le cas d'une implémentation d'un client virtuel, la mise à jour qui est distribuée par le serveur sera automatique et la même sur tous les clients ce qui facilite la maintenance car c'est le distributeur qui est responsable de la mise à jour et non le client.
- Mobilité et accès au réseau. La mise en place d'un réseau P2P permet de découpler l'accès à l'information et aux ressources du système. De ce fait, les clients peuvent travailler directement entre eux sans supervision après mise en relation et partager leurs ressources avec les autres clients qui en ont besoin. Le réseau peut évoluer sans que cela ait un fort impact sur la collaboration. Les clients peuvent être plus mobiles du fait de la grande disponibilité offerte par cette architecture à moindre coût.
- [NAT Traversal](#) et pare feu. Les applications traditionnelles de P2P comme BitTorrent ne permettent pas à deux pairs de communiquer directement lorsqu'ils sont derrière un [NAT](#). Grâce à l'utilisation du protocole [Interactive Connectivity Establishment \(ICE\)](#) les appareils peuvent atteindre plus de pairs, augmentant la vitesse d'échange.

Cette liste est un point de départ pour créer un service de virtualisation de clients pour le partage de données (3D) avec WebRTC. La mise à disposition volontaire de ressources (calcul, mémoire) en partage sur le réseau permet d'une part la coopération entre personnes afin de résoudre des problèmes nécessitant un haut degré de computation et d'autre part l'utilisation de ressources qui ne seraient pas ou sous utilisées.

L'optimisation de l'utilisation des ressources peut être effectuée avec WebRTC pour faire des grilles de calcul comme dans `browserCloud.js` [[Dias, 2015](#)] ou pour faire des analyses visuelles collaboratives dans un environnement hétérogène [[Li et al., 2015](#)].

En 2001, le standard [Data Distribution Service \(DDS\)](#) est un standard machine-à-machine massif, en temps-réel, hautement performant avec un système d'échange de données interopérables. [DDS](#) s'adresse principalement à des problématiques d'échanges financiers, de contrôle aérien, et de réseau électrique intelligent (*smart grid*). Il a fortement été promu pour mettre en place des applications liées à l'internet des objets. Les spécifications proposent deux niveaux d'interfaces. Le premier se

concentre sur la mise à disposition d'un système PubSub bas niveau centré données pour permettre la livraison efficace de la bonne information au bon destinataire. Le second, niveau optionnel, est une couche de reconstruction locale de la donnée permettant une intégration plus simple de DDS au sein d'une application. DDS est donc un intergiciel réseau basé sur une architecture PubSub qui gère la livraison de messages sans nécessiter l'intervention d'un utilisateur. Il détermine qui doit recevoir les messages, où sont situés les destinataires et ce qu'il se passe si un message n'est pas délivré. En cela, DDS permet une gestion plus fine de la qualité de service notamment concernant les paramètres de découverte des pairs.

2.3.3 Domain Driven Design

DDD (ou Conception Pilotée par le Domaine) est une approche de développement logicielle qui a pour objectif de définir une vision et un langage partagé (*ubiquitous language*) pour les personnes impliquées dans la construction d'une application [Evans, 2003]. 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.

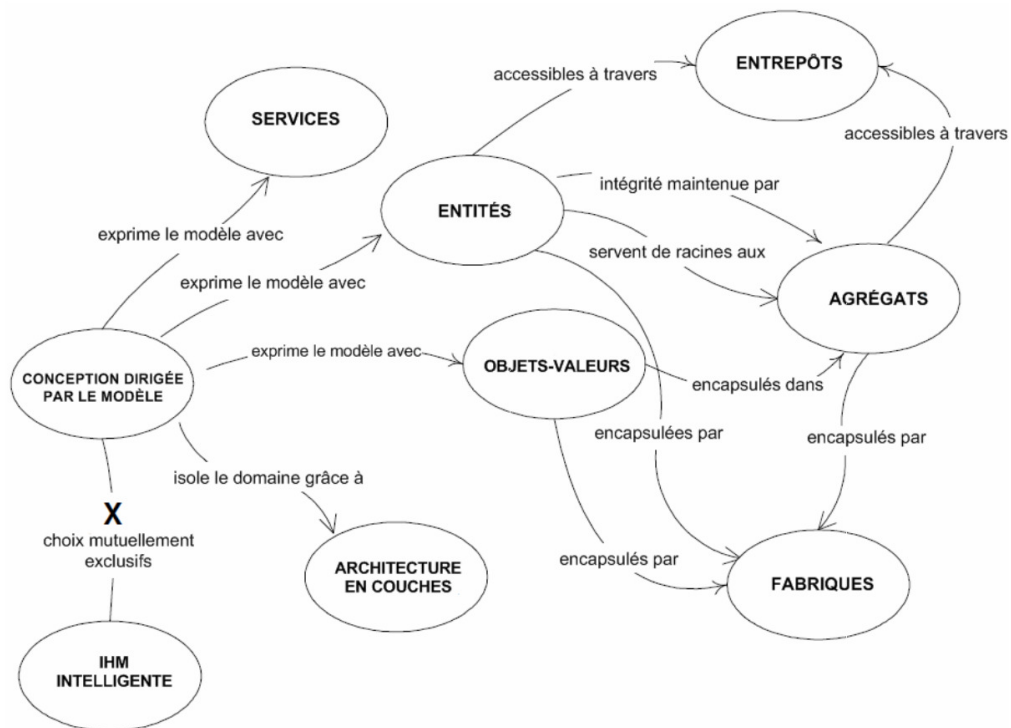


FIGURE 2.4 – Illustration du patron DDD et de ses artefacts (issue de [?]) sous licence Creative Commons)

En DDD, il existe des artefacts qui permettent d’exprimer, créer et stocker un modèle lié à un domaine (voir Figure 2.4). 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 maillages qui sont réutilisés alors un 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 modifiés car les utilisateurs ne sont intéressés que par les informations qu’elles portent.
- L’**agrégat** (*aggregate*) : une collection d’objets qui sont liés ensemble par une entité commune (*root entity*), connue sous le nom d’agrégat souche (*aggregate root*). Ce dernier garantit la cohérence des modifications faites au sein de

l'agrégat en interdisant aux objets externes de faire référence à ses membres. Par exemple, dans une modélisation 3D collaborative un utilisateur ne peut pas modifier le nom d'un autre utilisateur. Un maillage n'a également pas d'intérêt à connaître les informations des utilisateurs, c'est la scène qui gère l'interface entre les maillages et les utilisateurs.

- **L'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 [Evans, 2003] et [Vernon, 2013]⁸.

Le **DDD** a également l'avantage de fonctionner en harmonie avec les principes de l'**Event Sourcing (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 Responsibility Segregation

Une solution architecturale courante en **DDD** contient quatre couches : l'interface utilisateur (présentation), la couche application (coordination de l'activité de l'application), la couche domaine (cœur du logiciel métier), la couche infrastructure (interface entre les couches, persistance des objets métier...). La Figure 2.5 montre que ces quatre couches s'accordent bien avec l'architecture **Command Query Responsibility Segregation (CQRS)** qui sépare le flux d'écriture et de lecture dans le logiciel.

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

Introduit par Greg Young en 2009 [Young, 2009], **CQRS** est un patron de conception qui repose sur le principe de séparation des composants de traitement métier de l'information (écriture) et de la restitution de l'information (lecture). Le cadre offert par ce principe permet de lever certaines contraintes d'architecture comme 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 par qui est ce possible.

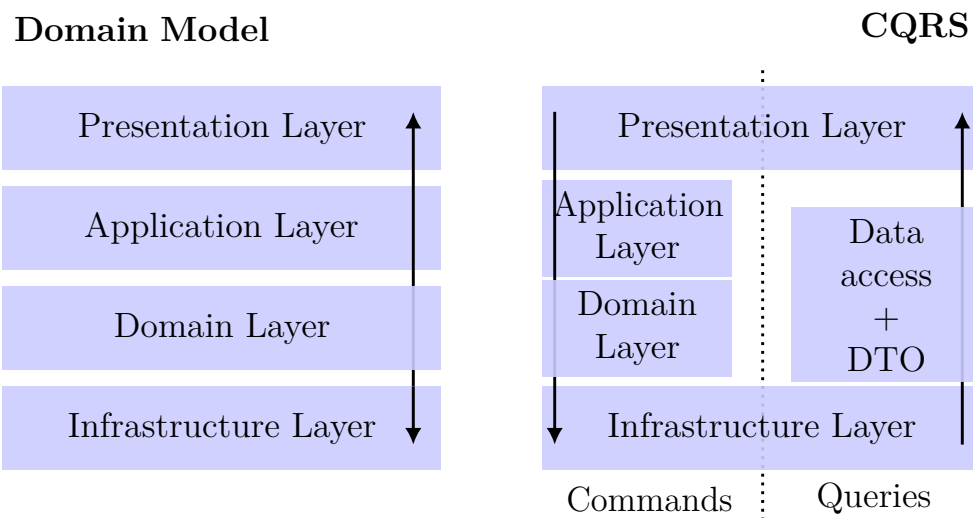


FIGURE 2.5 – Architecture en 4 couches du DDD (gauche) en miroir avec l'architecture CQRS (droite)

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

- En traitement/ écriture : besoins transactionnels, garantie de cohérence (*consistency*) des données, de normalisation.
- En consultation/lecture : dénormalisation, scalabilité.

La plupart des architectures en couches ne font pas explicitement référence à ces problèmes. Le fait de tout sauvegarder dans une base de données centralisée peut être une étape dans la gestion de la collaboration mais l'altération des données est souvent exacerbée par l'utilisation de caches comme accélérateur de performance.

L'**immuabilité** en **CQRS** est un concept clé qui prévient la modification de l'état interne d'une commande ou d'un événement. Les commandes sont immuables car leur usage nécessite un envoi direct au domaine pour être traitées. Quant aux événements, ils sont immuables car ils représentent ce qui s'est produit dans le passé (qu'on ne peut donc pas changer).

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

2.3.5 Event Sourcing

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

L'**ES** est souvent présent dans des architecture asynchrones qui ont l'avantage de pouvoir utiliser des queues de message, plusieurs bases de données et où la partie lecture est éventuellement consistante. La plupart de la littérature concernant ce patron se trouve en ligne, dans des billets de blog, des présentations, ou de la documentation logicielle. La littérature académique est relativement réduite, 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'événements capture tout ce qui est nécessaire à la reconstruction de l'état courant' ». Il voit les événements comme immuables et le journal d'événement (*event log*) comme un stockage linéaire (*append only store*) des événements. Les événements ne sont jamais supprimés, le seul moyen de l'"annuler" consiste à effectuer générer un événement rétroactif. Une fois qu'un événement rétroactif est ajouté, l'événement rétroactif agit à l'inverse de l'événement précédent pour

compenser ses effets. Dans ce billet, Fowler n'établit pas clairement la distinction entre les événements et les commandes qui déclenchent ces événements. Ce problème est considéré dans plusieurs travaux.

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

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

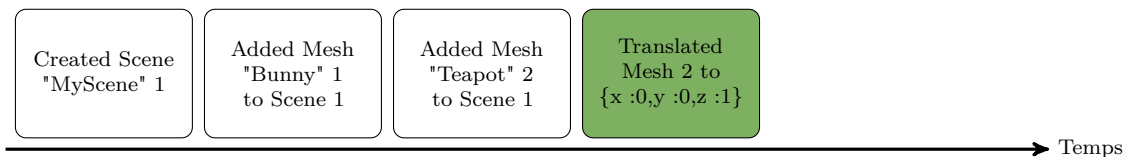


FIGURE 2.6 – Transaction en Event-Sourcing

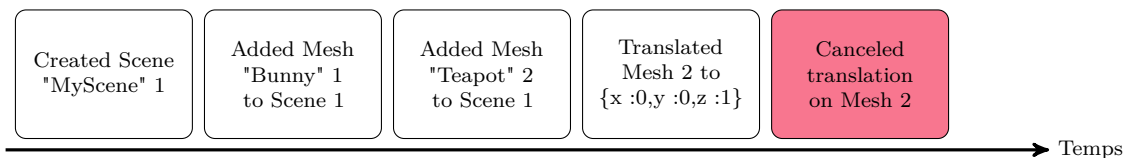


FIGURE 2.7 – Transaction avec compensation en Event-Sourcing

Les avantages et les inconvénients de l'approche ES sont orientés selon le cadre dans lequel elle est utilisée. En reprenant ceux cités par Klamer [Klamer, 2013], on peut les reprendre 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 système critiques, les applications d'Informatique

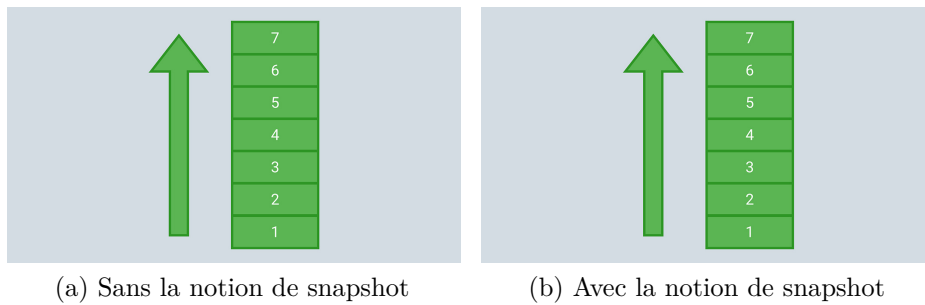


FIGURE 2.8 – Snapshot en Event-Sourcing

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 événements existent déjà donc les données sont déjà disponibles et ont seulement à être analysées (dans notre exemples, les informations sont inhérentes aux événements). De plus, les événements étant stockés depuis le début de la vie de l'application, beaucoup plus de données peuvent être analysées. On pourra 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

Les performances de l'application peuvent être affectées par l'utilisation de l'ES. En effet, pour récupérer l'état courant à partir de l'Event Store, il est nécessaire de le calculer à partir des événements reçus. A chaque événement créé (et stocké), ce processus sera alourdi car ils ne peuvent être supprimés. Plus la pile d'événements est longue, plus cela prend du temps (linéaire). On peut considérer d'une part que l'on a besoin des événements que lorsque qu'une commande est effectuée. Un [Snapshot](#) est une capture de l'état de l'agrégat à un moment qui correspond à l'empilement d'événements ayant mené à cet état. En créant un [Snapshot](#), on évite de reconstruire un état à partir du début de la vie de l'agrégat ; on empile les nouveaux événements à partir de ce snapshot. L'utilisation du [Snapshot](#) est intéressante à partir du moment où la pile d'événements devient plus lourde que le [Snapshot](#) lui-même. Il est donc important de bien déterminer la périodicité du déclenchement du [Snapshot](#) (temporelle ou selon le nombre d'événements).

Le parcours des événements se fait classiquement du premier au dernier (*bottom-up*). Si on utilise les [Snapshot](#), on peut se permettre de les parcourir dans l'autre sens (*top-bottom*) jusqu'à trouver un [Snapshot](#) puis appliquer tous les événements qui sont arrivés entre ce [Snapshot](#) et l'état courant. 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 événement passé. Le fait de travailler avec des [Snapshot](#) crée une dépendance des snapshots qui doivent intégrer les modifications du domaine. Une solution est de recalculer les snapshots quand le domaine est modifié mais cela reste coûteux (et à éviter).

Event Sourcing versus Command Sourcing

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

Un événement représente quelque chose qui est arrivé dans le domaine. Un événement appliqué sur un état, donne un nouvel état. La condition pour appliquer

un pur déterminisme en **ES** est la fonction suivante : $\text{State} \rightarrow \text{Event} \rightarrow \text{State}$. Cette fonction met en correspondance les (mêmes) entrées avec les (mêmes) sorties, on peut 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) : $\text{State} \rightarrow \text{Command} \rightarrow \text{Event list}$

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'accroît lors l'interaction avec des systèmes externes. L'aspect fonctionnel de l'application de changement d'état de l'**ES** a l'avantage de permettre de reconstruire l'état sans 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 de 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.

2.3.6 Bilan

parler de
la concu-
rence

Cette section présente différentes approches pour intégrer une architecture orientée évènements dans un système collaboratif respectant les règles liées au métier. 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 participants. Le mécanisme d'abonnement autorise une grande variété de stratégies quant au routage des messages.

Dans un second, différents patrons de conceptions sont présentés. Le **DDD** est plutôt considéré comme un patron de conception stratégique. Ses principes permettent d'avoir 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. Cela a des impacts à différents niveaux. 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. Côté écriture, le contenu des commandes est validé tout au long de la partie commande, notamment avec l'intégration des règles métiers dans le modèle du domaine. L'**ES** est le patron qui introduit la notion d'évènement au sens que « ce qui s'est passé ». Les évènements sont orientés métiers en suivant les lignes du **DDD**. Le flux d'évènements qui est généré par **CQRS** est stocké dans un **event store** qui correspond à la source de vérité de l'application. On peut connaître l'historique de l'application juste en regarde la pile d'évènements autant que recrée l'état de l'application à partir de l'**event store**.

2.4 Conclusion

Ce chapitre présente en trois sections les Dans les 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.

finir en
parlant
du P2P

Chapitre 3

Contributions scientifiques

Contents

3.1	Introduction	50
3.2	Modèle évènementiel pour l'intégration du domaine 3D dans les EVC	50
3.2.1	Modèle général	53
3.2.2	Mécanisme de gestion de version	56
3.2.3	Cohérence Éventuelle en CQRS	56
3.2.4	Potentielles applications et autres utilisations	57
3.2.5	Bilan	58
3.3	Architecture de communication hybride	58
3.3.1	Présentation générale	61
3.3.2	Event Store distribué	62
3.3.3	Persistance à long terme	64
3.3.4	Synchronisation client-serveur	64
3.3.5	Gestion de la cohérence	65
3.3.6	Bilan	65
3.4	Conclusion du chapitre	65

3.1 Introduction

Nous avons vu que 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 à au rapport des données à la 3D dans le contexte de l'assemblage d'objets 3D dans une scène partagée cela permet de mieux comprendre les fonctionnalités spécifiques à la 3D (transformations), l'historisation des données et leur visualisation. Les ressources nécessaires à la collaboration demandent une haute disponibilité et une forte réactivité. Ces besoins impliquent de mettre en place 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, on introduit 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, on 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. C'est pourquoi la seconde contribution s'intéresse à l'architecture de communication dans un environnement web. 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

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

La méthodologie orientée événements intègre plusieurs aspects souvent laissés de côté dans la littérature concernant le développement les environnements virtuels

collaboratifs pour la visualisation et la manipulation d'objets 3D. Par exemple, elle a l'avantage de proposer un couplage lâche. Dès lors, la réutilisation des données produite lors de la collaboration peut facilement être réutiliser 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 [Chandy *et al.*, 2011]. L'EP intègre plusieurs fonctionnalités :

- 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 situation critiques qui nécessitent une réponse
- Déterminer la réponse la plus adaptée à ces situations
- Surveiller (*monitor*) l'exécution de cette réponse

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

Constat

Par nature, une architecture orientée évènements 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 l'évènement va subir par la suite ou qui cela va concerner. C'est pourquoi les architectures orientées évènements 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 on reste dans un cadre et un domaine connu.

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 architectures orientées événements repose le plus souvent sur communication client-serveur pour faciliter la gestion des données dans le système distribué. C'est pourquoi l'exploitation du patron de conception [CQRS](#) a originalement été développé avec cette [CS](#) qui sont toujours connectées (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, on peut donc se permettre 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 important 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 échangeant réseaux sont limités car qui sont très coûteux d'un point de vue énergétique pour les appareils mobiles [[Koskela et al., 2015](#)]. De plus, l'utilisation de la bande passante est onéreuse et parfois limitée voire inexistante, il est donc nécessaire de tirer parti de tous les appareils participant à la collaboration au lieu de tout faire reposer sur le serveur. Chaque appareil participant à la collaboration doit être autonome et le plus indépendant possible en termes de ressources (données, réseaux, validation experte).

Contribution

Pour répondre à cette problématique, nous proposons 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 (travaille hors ligne) ou involontaire (coupure). On garantit ainsi à l'utilisateur l'utilisation du système avec un historique performant où chaque connexion est l'occasion de mettre le système à jour.

3.2.1 Modèle général

La composition de l'architecture s'est effectuée avec en arrière pensée les lignes directrices énoncées plus haut [Xhafa et Poulouvassilis, 2010]. Utiliser une architecture orientée évènements pour faire de la modélisation 3D peut sembler non nécessaire. Cependant lorsqu'on s'intéresse aux apports que cela peut générer pour tous les métiers engagés (utilisateurs, développeurs, analystes métier) on peut se demander pourquoi cela n'a pas été plus mis en avant auparavant. La sensibilisation à l'historique des données et aux interactions inter-utilisateurs est partagée par les utilisateurs et les analystes métier. Quand à la sensibilisation à la distribution des données elles est une composante importante pour les utilisateurs et les développeurs. Concernant les premiers, cela garantit une certaine autonomie dans la création. Pour les seconds, 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.

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 étendues 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 événements à partir du journal d'événements pour mettre à jour la base de données. L'état interne dérivé à partir des événements consommés est un modèle de lecture en mémoire et contribue à la partie requête (Q) du CQRS.

Processeur Un processeur est un acteur qui consomme des événements à partir d'un journal d'événements et produit les événements traités pour un autre journal d'événements. Les processeurs peuvent être utilisés pour connecter les journaux d'événement au traitement des événements..

Les événements produits par un des composants présentée ci-dessus peuvent être consommés par d'autres de ces abstractions s'ils partagent un journal d'événements local ou distribué.

Un **journal d'événements** peut fonctionner sur un seul site ou être répliqué sur plusieurs sites. Le site est considéré comme une zone disponible qui accepte l'écriture d'un journal d'événements local même s'il est partitionné sur plusieurs sites. Les journaux d'événements locaux situés sur plusieurs sites peuvent être connectés par le biais d'un journal d'événements dit « répliqué » qui a pour responsabilité de préserver l'ordre causal des événements.

Les sites peuvent être situés à des endroits géographiquement distincts ou sur des nœuds à l'intérieur d'une même grappe (*cluster*) ou encore être sur le même nœud mais traités séparément selon les zones disponibles nécessaires au fonctionnement de l'application. Les Acteurs et les Processeurs écrivent cependant toujours sur leur journal d'événements local. Les composants peuvent soit collaborer sur un journal d'événements local sur le même site, ou bien au travers d'un journal répliqué sur différents sites.

Il est important de différencier le journal d'événements de la base de données (côté serveur) ; la base de données peut ne contenir qu'une partie du journal. Une base de données peut également être considérée comme un élément complémentaire au journal d'événements, cependant et bien que parfois confondus, ils restent bien distincts conceptuellement.

Le journal d'événements commun est la base des échanges pour communiquer par le biais d'événements de collaboration. Ce type d'architecture se retrouve dans différents cas d'utilisation :

- *Processus métier distribués*. Les acteurs de différents types utilisent des événements pour communiquer et parvenir à résoudre un problème commun. Bien qu'ils jouent des rôles différents dans le processus métier, ils réagissent

à la réception d'évènements (*reactive programming*) 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 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 de manière asynchrone sur les autres des journaux d'évènements des sites distants. 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 réplifications. 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.

3.2.2 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 évènements. Ces évènements sont considérés comme « soumis » (*uncommitted*) mais pas encore « publiés » (*committed*). Pour qu'ils le deviennent, l'agrégat concerné par ces évènements 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 la valeur initiale choisie pour les agrégats du *framework* (on peut commencer à n'importe quelle valeur $-1, 0, \dots$).

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 2 phase commit ou de protocoles Paxos (consensus) complexifiant les échanges. LaCE introduit l'idée que toutes les répliques se réconcilient au bout d'un moment (*forward progression*) pour avoir le même état final. Si le caractère vicié d'une information est détecté, le système doit le « réparer » pour obtenir le bon état.

L'ordonnancement des événements durant les mises à jour reste identique lorsque les événements sont rejoués par la suite car l'ordre est issu de l'ES est déterminée par l'ordonnancement des événements stockés localement. Au sein d'une réplique, tous les composants CQRS respectent cet ordonnancement. L'ordre de stockage des événements répliqués sur des sites distincts est cohérent à cause de l'ordre causal : les événements liés causalement ont le même ordre sur tous les sites alors que les événements concurrents peuvent avoir un ordre différent. Cette propriété est importante pour obtenir une cohérence causale (forte) dans une application qui respecte le théorème Consistency, Availability, and Partition Tolerance (CAP).

3.2.4 Potentielles applications et autres utilisations

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

GIT-like app Les solutions pour faire de la gestion de version, comme Mesh-Git [Denning et Pellacini, 2013] 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 événements 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, on peut 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. Si on choisit d'enregistrer le chemin de la caméra et les actions utilisateur comme des événements que l'on considère comme 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 monitoring 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, on va pouvoir lever une alerte facilement.

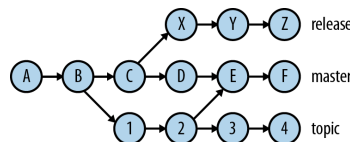


FIGURE 3.1 – Exemple d'arbre de commits Git

3.2.5 Bilan

3.3 Architecture de communication hybride

Dans la section précédente, l'introduction du modèle orienté évènement 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 ...

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 [Sun *et al.*, 1998] (cf Section ??). 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 conflit en 3D peut vite devenir écrasante. 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

WebRTC est une technologie qui fournit aux navigateurs *desktop* et mobiles la possibilité de communiquer en temps-réel (*Real-Time Communications*) via une collection de standards, protocoles et APIs JavaScript. Le standard WebRTC est développé au sein du W3C et de l'Internet Engineering Task Force (IETF) depuis 2011 dans sa version 1.0 (Working Draft). 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 3.2) qui lui offre des fonctionnalités similaires (fiabilité, ordonnancement, sécurité).

La jeunesse du protocole (2011) fait que peu de travaux académiques l'utilisent pour créer des EVC 3D [Desprat *et al.*, 2015, Steiakaki *et al.*, 2016]. Côté commer-

cial, 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)¹, des systèmes de partage de fichiers² et autres applications (capture d'écran, réalité augmentée, jeux)³.

La simplicité de la mise en place (très semblable à WebSocket) et la richesse apportée par le P2P n'y sont pas étrangers. Les réseaux P2P ne sont souvent pas bien connus des développeurs web qui sont habitués aux architectures client-serveur, ce qui peut représenter une barrière d'entrée pour de nouvelles applications réseau. Dans les systèmes P2P, il n'y a pas de fermes de serveurs comme il peut en exister dans les applications client-serveur. Les machines clientes et les systèmes d'extrémité (*end system*) sont considérée 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. Cependant, lors d'un déploiement P2P réel, des serveurs sont en général utilisés pour télécharger l'application cliente qui contient la couche intergicielle P2P et fournit à l'utilisateur le service de connexion nécessaire à l'accès au service. Ces serveurs sont moins coûteux que peut l'être une ferme de serveurs. De plus, pour beaucoup de services décentralisés, il n'existe pas de solution pérenne pour tester ces applications à l'échelle ; chaque service doit créer son propre système de test. Ce contexte technologique est une motivation supplémentaire dans les travaux de cette thèse.

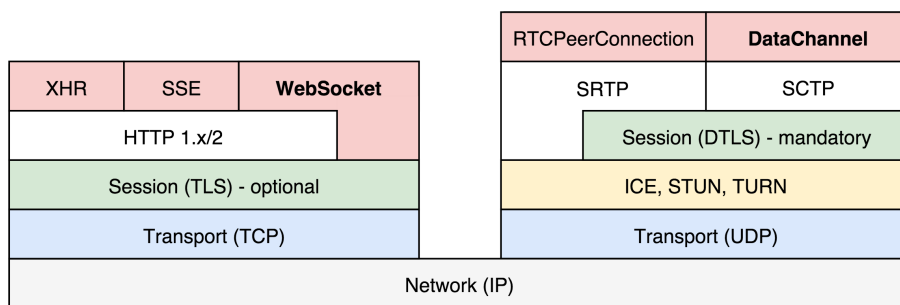


FIGURE 3.2 – Pile des protocoles IP : TCP vs UDP

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

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

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

Contribution

Une architecture de communication hybride en trois parties serveur, persistance, pairs est utilisée dans 3DEvent [Desprat *et al.*, 2016]. Cela permet de concilier à la fois les avantages d’une architecture client-serveur et ceux du P2P en évitant certains désavantages occasionnés dans ceux-ci (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 de 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 3.3. 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’envoi et de réception de message. Les différents message vont concerner la partie routage de l’information, le maintien à jour du répertoire de voisin et leur dynamique, 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 information de contenu à traiter qui correspond à stockage de contenu. Cette dernière 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 actif sera configurer pour traiter et relayer les données tandis qu’un pair passif ne fait que stocker et relayer les données.

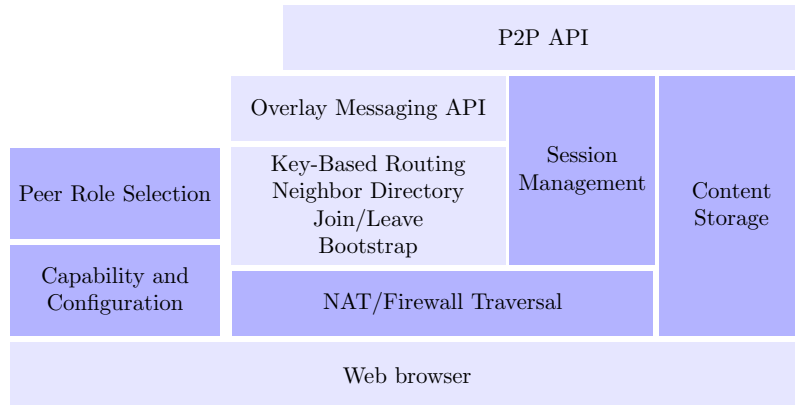


FIGURE 3.3 – Composants de chaque pair dans 3DEvent (point de vue réseau)

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 3.4 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 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, gérée par le gestionnaire d’instance, et retourne la liste des pairs avec lesquels le pair doit se connecter. Pour chaque pair *idB* de la liste 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 Event Store distribué

L’Event Store est un composant clé dans le traitement des événements. Il prend en entrée des événements générés ou reçus extérieurement et produit en sortie des événements cohérents qui peuvent être par la suite publiés. Les événements 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

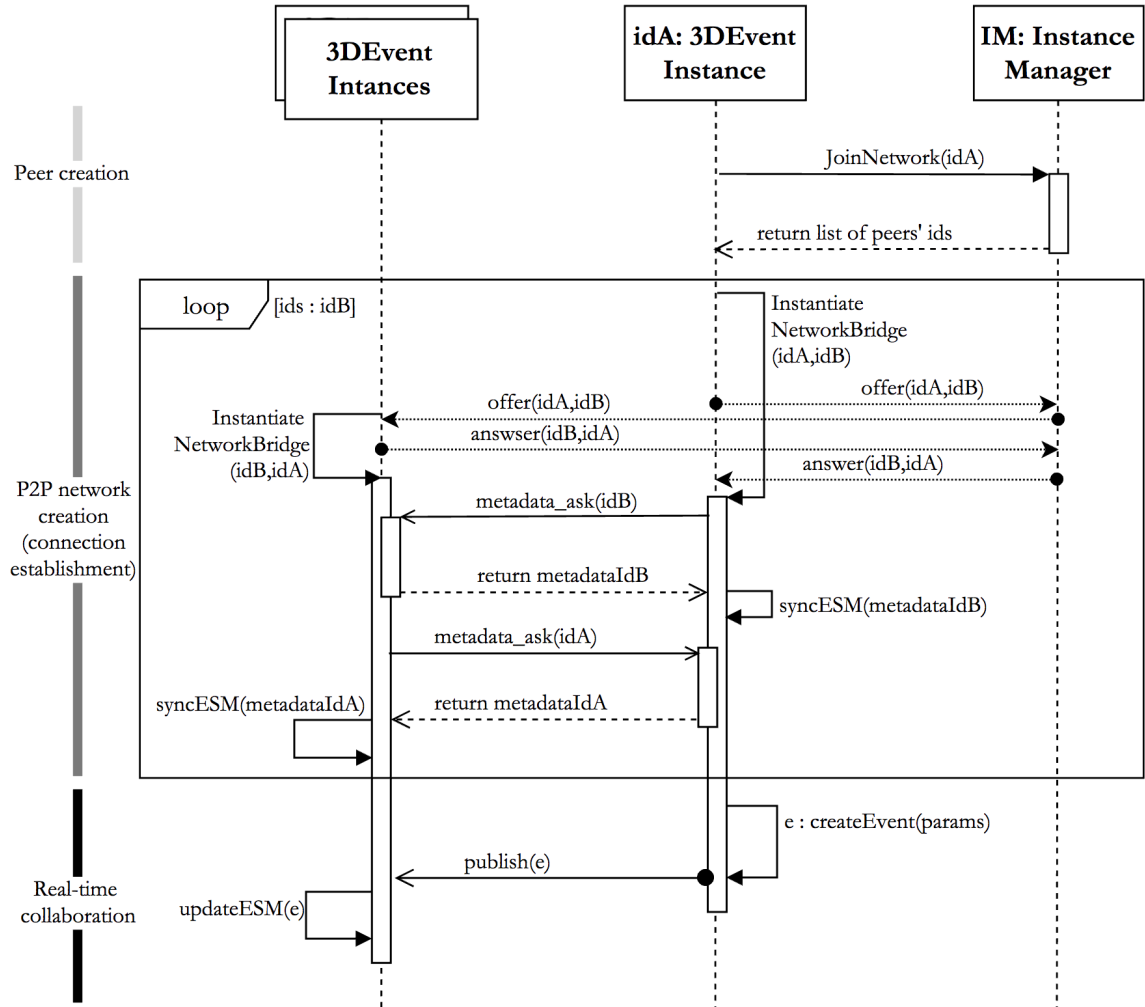


FIGURE 3.4 – Protocole de connexion au réseau d'instance 3DEvent

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'événements. Un flux d'événement est représenté par un tableau d'événements indexés en séquence permettant de stocker les événements 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 événement de l'instance courante, l'ESM récupère (ou crée) le flux d'événements associé à l'agrégat référencé par l'événement. Alors, la cohérence de la version est vérifiée en comparant la version attendue (exposée dans les métadonnées de l'événement) et la version actuelle de l'agrégat. Si les deux numéros de versions sont égaux, l'événement est ajouté à la fin 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'événement est publié.

3.3.3 Persistance à long terme

Dans un contexte de collaboration sur le long terme pour des entreprises où le besoin d'information accessible sur le long terme est nécessaire. La persistance 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 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 Gestion de la cohérence

Respect de la causalité

Convergence des répliques

Préservation de l'intention

3.3.6 Bilan

3.4 Conclusion du chapitre

Dans ce chapitre nous avons vu les différents composants de l'architecture de communication pour 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 une récupération du contenu rapidement et octroie une autonomie certaine aux contributeurs.

Chapitre 4

Implantation

Contents

4.1	Introduction	68
4.2	3DEvent : Plateforme web de manipulation collaborative d'objets 3D	68
4.2.1	Éditeur 3DEvent	68
4.2.2	Interface utilisateur	69
4.2.3	Flexibilité de la visualisation	70
4.3	Intergiciel P2P pour l'échange de données 3D	71
4.3.1	Données d'échange	72
4.3.2	Synchronisation des données	72
4.4	Résumé des choix techniques	74
4.5	Bilan	74

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 événement pour répondre à des contraintes liées 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 (événements) 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 de point de vue d'un collaborateur, se mettre à sa place, est une manière de comprendre son point de vue, son fonctionnement et d'imaginer ses perspectives de conception à travers le point de vue qu'il a choisit.

4.2.2 Interface utilisateur

Dans le but de proposer une IU proche des fonctionnalités métier liées à la modélisation 3D avec une interface orientée tâche. De cette manière, le modèle orienté événements présenté dans le précédent chapitre est directement orientée 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 événements qui ont eut 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

1. Un rôle peut être défini par le biais du [framework](#) 3DEvent

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.

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'événements. Pour Abdullin, «la projection est le processus de conversion (ou d'agrégation) d'un flux d'événement en une représentation structurée. 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 événements, ne laissant filtrer que ceux qui sont pertinents pour la vue. La projection fournit une vue adaptée (filtrée, enrichie...) du flux d'événements au client. Elle peut également être utilisée pour mettre en avant des aspects experts (notifications, déclenchement d'action) ou des raisons de confidentialité. Une projection peut être créée de manière synchrone (à la volée) au fur et à mesure de la publication des événements ou de manière asynchrone et donc découplée du flux des événements.

Du fait de la nature d'un réseau [P2P](#), les pairs ne reçoivent pas forcément les paquets réseau de manière ordonnée. Par conséquent, les messages pouvant arriver dans n'importe quel ordre, qu'arriverait-il si un événement A (eA) nécessitant un autre événement 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 événements (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

événements qui arrivent. Par exemple, si on essaie d'ajouter un objet dans une scène (eA) sans avoir créé la scène (eB) la projection met en attente eA jusqu'à recevoir eB . Dans le cas où eB n'arrive jamais, la projection ne pourra jamais utiliser eA .

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 compatibles avec les besoins liés à la 3D, le web et la collaboration.

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

La topologie de l'architecture de communication repose sur la mise en relation automatique des clients par le biais du serveur pour établir une connexion [WebRTC](#). Pour ce faire, chaque client envoie son identifiant (ID) lors de sa première requête au serveur qui le stocke. Selon le paramétrage de la connectivité directe minimum établie préalablement, le serveur recherche aléatoirement l'ID d'autres clients qui satisfont la règle de connectivité. Cette règle de connectivité minimum permet d'ajuster la densité du maillage (connectivité élevée : maillage partiel dense voire complet ; connectivité faible : maillage partiel éparse) et d'obtenir une topologie maillée adaptée aux besoins de l'application en termes de synchronisation (temps-réel ou pas) 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 pair d'échanger des données arbitraires avec d'autres à partir du navigateur avec des propriétés de livraison personnalisables (fiable ou non fiable, ordonné ou non ordonné) selon le transport sous-jacent. 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, il est lisible et interopérable. Le format de fichier [GL Transmission Format \(glTF\)](#) se base sur la représentation [JSON](#) afin de décrire une scène 3D. Ce type de données est assez abstrait et suffisamment générique pour représenter n'importe quelle donnée. Par exemple le format de fichier [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.

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

Tableau 4.1 – 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

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 les mises à jour générées par les clients entre les clients 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 évènements 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 requière 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 mode est principalement utilisé pour la lecture de vidéo en ligne. En comparaison le mode téléchargement est moins restrictif et relativement simple à mettre en place. Tout comme les systèmes utilisant une architecture client-serveur, la transmission de contenu en P2P peut également être catégorisée selon ces deux modes. Une catégorisation plus précise du flux continu peut être donnée selon quand le contenu est généré : en direct (live) et à la demande (*on-demand*).

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

Tableau 4.2 – 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

4.4 Résumé des choix techniques

Base de données NoSQL L'évolution de l'utilisation du web en tant que plateforme applicative a encouragé le changement dans le stockage des données pour de nouveaux besoin supportant de larges volumes de données (comme les données 3D). Une base de données [Not Only SQL \(NoSQL\)](#) fournit un schéma libre et dynamique ainsi qu'une API de requête riche pour la manipulation de données. De plus, la possibilité d'enrichir un document à la volée facilite l'évolution des objets (3D) et la maintenance de l'application. 3DEvent intègre un système de persistance sur le long terme caractérisé par une base de données [NoSQL](#).

La base de données ([NoSQL](#)) conserve tous les événements qui 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	76
5.2	Cas d'étude : Assemblage collaboratif d'objets 3D dans un environnement web	76
5.3	Expérimentation 1 : preuve de faisabilité	76
5.3.1	Présentation de l'expérimentation	76
5.3.2	Résultats	76
5.3.3	Discussion et Conclusion	76
5.4	Expérimentation 2 : Intégration du framework évènementiel	76
5.4.1	Résultats et Discussion	79
5.5	Comparaison entre l'expérimentation 1 et l'expérimentation 2	81
5.5.1	Résultats	81
5.5.2	Discussion et Conclusion	81
5.6	Bilan	81

5.1 Introduction

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

Dans [Desprat *et al.*, 2015] et [Desprat *et al.*, 2017],

5.3 Expérimentation 1 : preuve de faisabilité

Cette expérimentation est tirée de l'article [Desprat *et al.*, 2015]. 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 [Desprat *et al.*, 2017] 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 (*monitoring*) 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 positionnés (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'évènements générés et l'horodatage de chaque évènement 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 à devaient remplir un questionnaire basé sur l'expérimentation pour exprimer leurs retours qualitatifs concernant leur expérience (Annexe ??) .

Le questionnaire est inspiré de [Lewis, 1995], qui permet d'évaluer la facilité d'utilisation du système et l'implication dans la collaboration de chacun des participants. En utilisant une échelle de notation sur sept points [Lewis, 1993], (1 : pas d'accord ; 7 : d'accord) pour avoir assez de points de discrimination.

Au cours des différentes sessions collaboratives, l'application a 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 collaboratives. 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 d'un point de vue 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 exceptions. 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 interagi (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 revienne 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 nombreuses 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 collaboration 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'évènement 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 n'est donc pas en reste malgré quelques aspects à améliorer. Rappelons que bien que la complexité des modèles ne soit 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 médium (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 ».

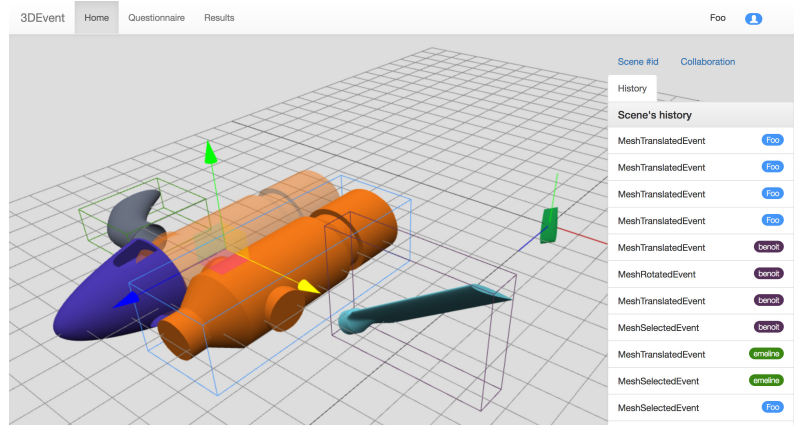
La satisfaction générale à propos de l'expérimentation et la satisfaction concernant la collaboration et l'expérience utilisateur montrent que les participants ont positivement apprécié faire de la modélisation collaborative 3D dans un navigateur web. Quant au fait que le nombre d'utilisateur améliore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

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

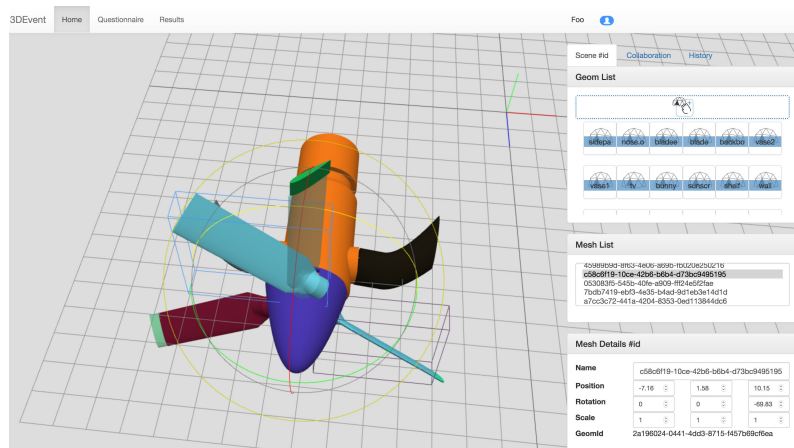
5.5.1 Résultats

5.5.2 Discussion et Conclusion

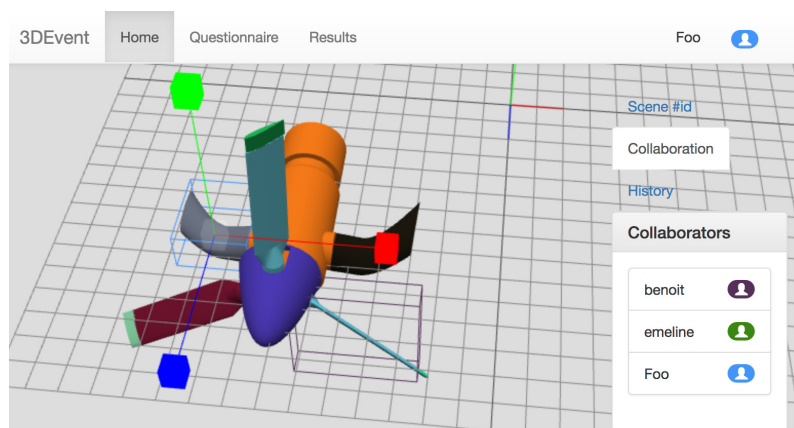
5.6 Bilan



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

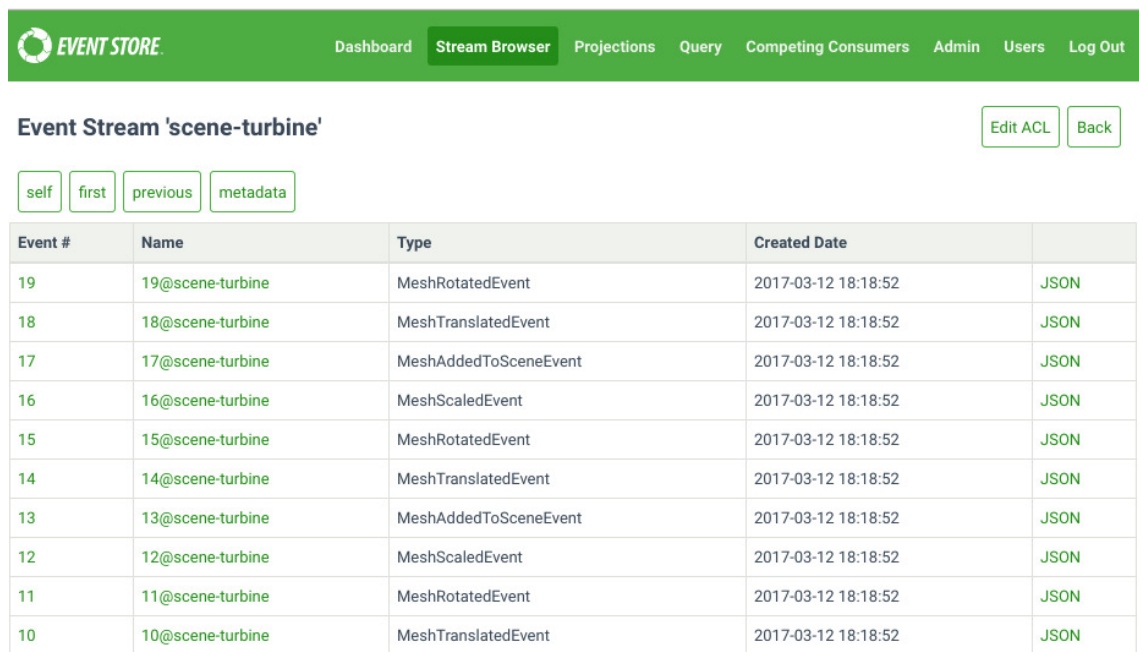


(b) Rotation (environnement 3D) et outils pour la manipulation d'objet 3D (panneau latéral)



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

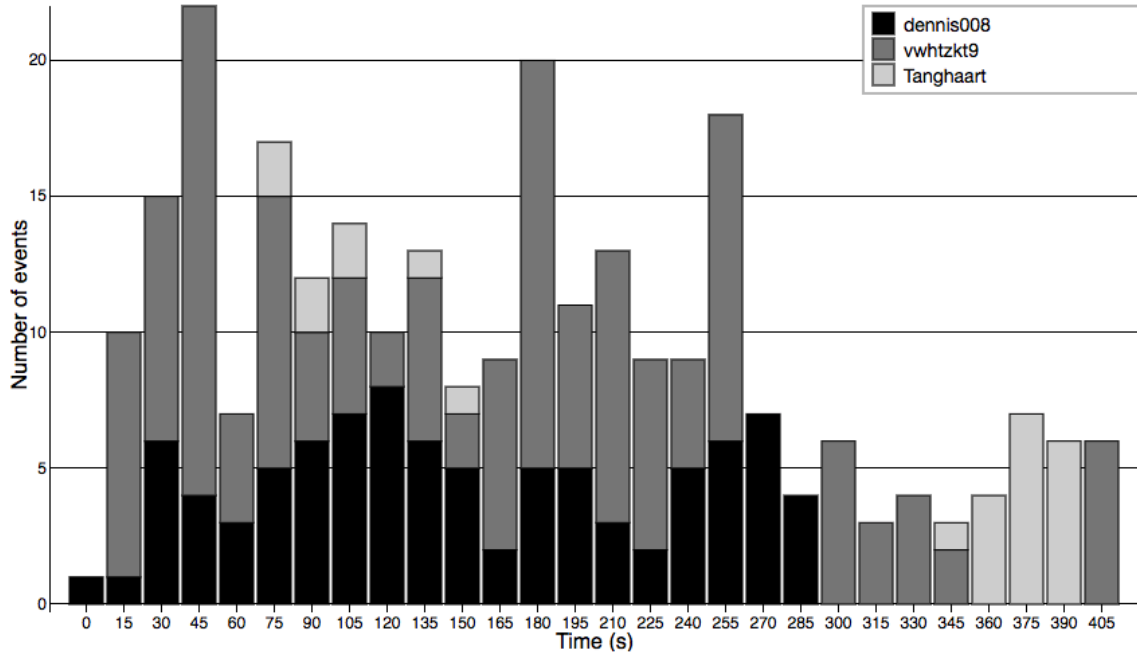
FIGURE 5.1 – Interface utilisateur pendant une session collaborative (trois personnes)



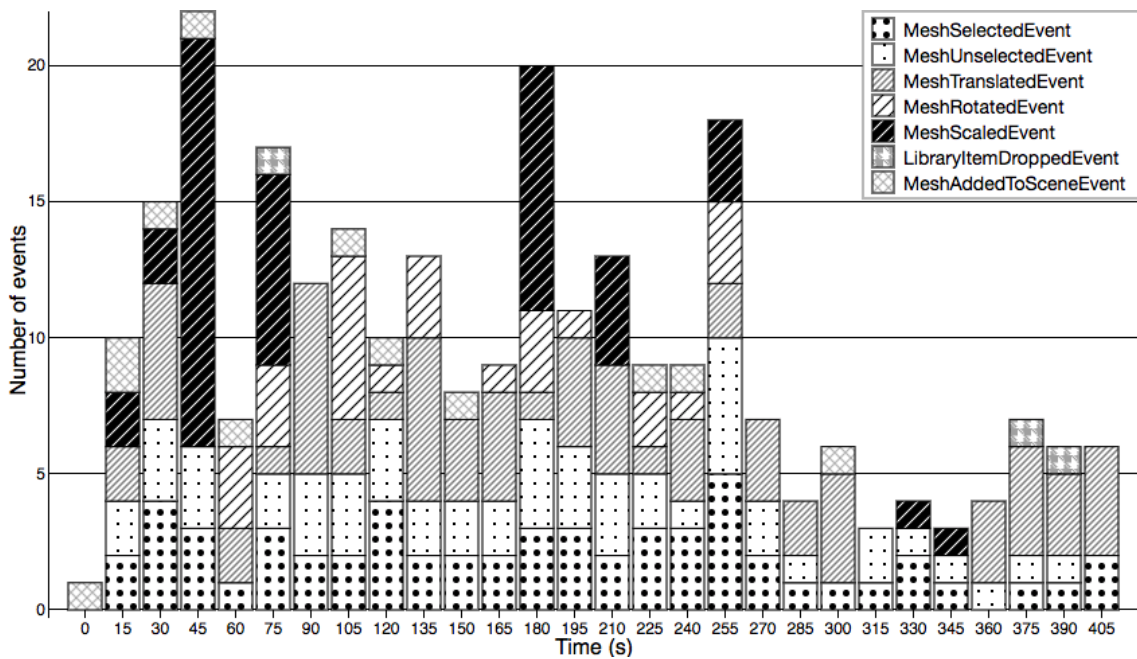
The screenshot displays the Event Store Stream Browser interface. At the top, a green navigation bar contains the 'EVENT STORE' logo and links for Dashboard, Stream Browser (selected), Projections, Query, Competing Consumers, Admin, Users, and Log Out. Below the navigation bar, the title 'Event Stream 'scene-turbine'' is shown, along with 'Edit ACL' and 'Back' buttons. A row of buttons includes 'self', 'first', 'previous', and 'metadata'. The main content is a table listing events in the stream, ordered from newest (19) to oldest (10). Each row includes the event number, name, type, created date, and format (JSON).

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



(a) Par utilisateur



(b) Par type d'évènement

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

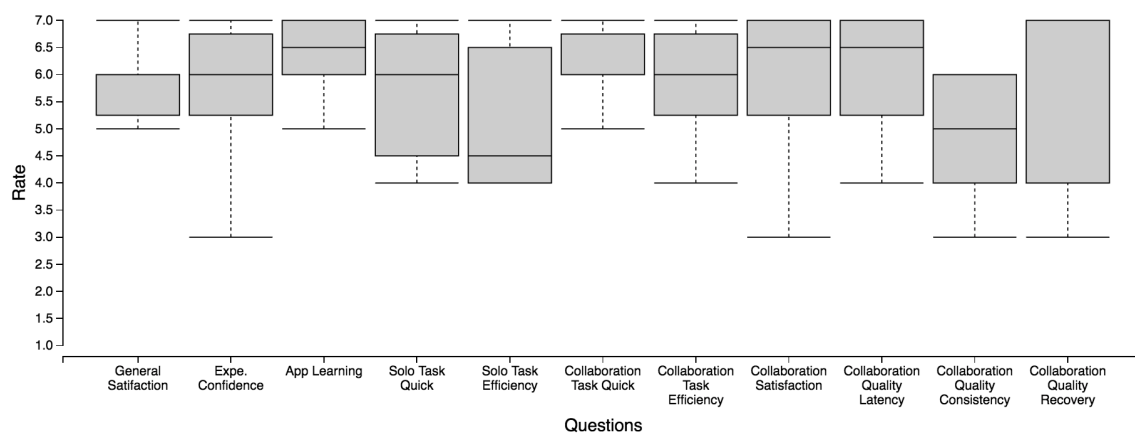


FIGURE 5.4 – Résultats des questionnaires collectés

Chapitre 6

Conclusion

Bibliographie

- [Banavar *et al.*, 1999] BANAVAR, G., CHANDRA, T., MUKHERJEE, B., NAGARAJA-RAO, J., STROM, R. et STURMAN, D. (1999). An efficient multicast protocol for content-based publish-subscribe systems. *Proceedings. 19th IEEE International Conference on Distributed Computing Systems (Cat. No.99CB37003)*, pages 262–272. [2.3.2](#)
- [Bang *et al.*, 2010] BANG, J. Y., POPESCU, D., EDWARDS, G., MEDVIDOVIC, N., KULKARNI, N., RAMA, G. M. et PADMANABHUNI, S. (2010). CoDesign : a highly extensible collaborative software modeling framework. *2010 ACM/IEEE 32nd International Conference on Software Engineering*, 2:243–246. [2.3.1](#)
- [Baran, 2015] BARAN, I. (2015). Under the Hood : How Collaboration Works in Onshape. [2.1.2](#), [2.2](#)
- [Becher, 2012] BECHER, M. (2012). Interactive Volume Visualization with WebGL. (06). [2.1.3](#)
- [Behr *et al.*, 2010] BEHR, J., JUNG, Y., KEIL, J., DREVENSEK, T., ZÖLLNER, M., ESCHLER, P. et FELLNER, D. W. (2010). A Scalable Architecture for the HTML5/X3D Integration Model X3DOM. *Proceedings of the 15th International Conference on Web 3D Technology*, 1(212):185–194. [2.1.3](#)
- [Bentley et Wakefield, 1997] BENTLEY, P. et WAKEFIELD, J. (1997). Generic Evolutionary Design. *Soft Computing in Engineering Design . . .*, pages 1–10. [1.1.3](#)
- [Birman et Joseph, 1987] BIRMAN, K. et JOSEPH, T. (1987). Exploiting virtual synchrony in distributed systems. *ACM SIGOPS Operating Systems Review*, 21(5):123–138. [2.3.2](#)
- [Brown *et al.*, 2003] BROWN, D., JULIER, S., BAILLOT, Y. et LIVINGSTON, M. (2003). An event-based data distribution mechanism for collaborative mobile augmented reality and virtual environments. *IEEE Virtual Reality, 2003. Proceedings.*, 2003. [1.1.3](#)

- [Calabrese *et al.*, 2016] CALABRESE, C., SALVATI, G., TARINI, M. et PELLACINI, F. (2016). cSculpt : a system for collaborative sculpting. *ACM Transactions on Graphics*, 35(4):1–8. [2.2](#)
- [Callahan *et al.*, 2008] CALLAHAN, S., SCHENK, M. et WHITE, N. (2008). Building a collaborative workplace. *Anecdote : putting stories to work*, pages 1–11. [1.1.1](#)
- [Carzaniga *et al.*, 2000] CARZANIGA, A., ROSENBLUM, D. S., SCIENCE, C. et WOLF, A. L. (2000). Achieving Scalability and Expressiveness in an Internet-Scale Event Notification Service. *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, pages 219–227. [2.3.2](#)
- [Carzaniga et Wolf, 2002] CARZANIGA, A. et WOLF, A. L. (2002). A Benchmark Suite for Distributed Publish / Subscribe Systems. *Program*. [2.3.2](#)
- [Castro *et al.*, 2002] CASTRO, M., DRUSCHEL, P., KERMARREC, A. M. et ROWSTRON, A. I. T. (2002). Scribe : A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, 20(8):1489–1499. [2.3.2](#)
- [Chandrasegaran *et al.*, 2013] CHANDRASEGARAN, S. K., RAMANI, K., SRIRAM, R. D., HORVÁTH, I., BERNARD, A., HARIK, R. F. et GAO, W. (2013). The evolution, challenges, and future of knowledge representation in product design systems. *Computer-Aided Design*, 45(2):204–228. [1.1](#)
- [Chandy *et al.*, 2011] CHANDY, M. K., ETZION, O. et AMMON, R. V. (2011). *The event processing manifesto*. Numéro 10201. [1.1.3](#), [3.2](#)
- [Cristea *et al.*, 2011] CRISTEA, V., POP, F., DOBRE, C. et COSTAN, A. (2011). Distributed architectures for event-based systems. *Studies in Computational Intelligence*, 347:11–45. [1.1.3](#)
- [Denning et Pellacini, 2013] DENNING, J. D. et PELLACINI, F. (2013). MeshGit. *ACM Transactions on Graphics*, 32(4):1. [3.2.4](#)
- [Desprat *et al.*, 2017] DESPRAT, C., CAUDESAYGUES, B., LUGA, H. et JESSEL, J.-P. (2017). Doctoral Symposium : Loosely Coupled Approach for Web-Based Collaborative 3D Design. In *Proceedings of ACM International Conference on Distributed Event-Based Systems*. [5.2](#), [5.4](#)
- [Desprat *et al.*, 2016] DESPRAT, C., JESSEL, J.-P. et LUGA, H. (2016). 3DEvent : A Framework Using Event-Sourcing Approach For 3DWeb-Based Collaborative Design in P2P. In *Proceedings of the 21st International Conference on Web3D Technology - Web3D '16*, pages 73–76. [3.3](#)

- [Desprat *et al.*, 2015] DESPRAT, C., LUGA, H. et JESSEL, J.-P. (2015). Hybrid client-server and P2P network for web-based collaborative 3D design. *WSCG 2015 Conference on Computer Graphics, Visualization and Computer Vision*, pages 229–238. [3.3](#), [4.3.2](#), [5.2](#), [5.3](#)
- [Dias, 2015] DIAS, D. (2015). browserCloud.js - A federated community cloud served by a P2P overlay network on top of the web platform. (May). [2.3.2](#)
- [Ellis et Gibbs, 1989] ELLIS, C. A. et GIBBS, S. J. (1989). Concurrency control in groupware systems. *ACM SIGMOD Record*, 18(2):399–407. [2.1.1](#)
- [Evans, 2003] EVANS, E. (2003). *Domain-Driven Design : Tackling Complexity in the Heart of Software*. Addison Wesley. [2.3.3](#), [2.3.3](#)
- [Gadea *et al.*, 2016] GADEA, C., HONG, D., IONESCU, D. et IONESCU, B. (2016). An architecture for web-based collaborative 3D virtual spaces using DOM synchronization. *2016 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, pages 1–6. [2.1.3](#)
- [Gotta, 2007] GOTTA, M. (2007). Categorizing Collaboration. [1.1.1](#)
- [Grasberger *et al.*, 2013] GRASBERGER, H., SHIRAZIAN, P., WYVILL, B. et GREENBERG, S. (2013). A data-efficient collaborative modelling method using websockets and the BlobTree for over-the air networks. *Proceedings of the 18th International Conference on 3D Web Technology - Web3D '13*, page 29. [2.1.2](#), [2.2](#)
- [Greenberg et Marwood, 1994] GREENBERG, S. et MARWOOD, D. (1994). Real time groupware as a distributed system. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work - CSCW '94*, pages 207–217, New York, New York, USA. ACM Press. [2.1.1](#)
- [Grimstead *et al.*, 2005] GRIMSTEAD, I. J., WALKER, D. W. et AVIS, N. J. (2005). Collaborative visualization : A review and taxonomy. *Proceedings - IEEE International Symposium on Distributed Simulation and Real-Time Applications, DS-RT*, pages 61–69. [2.1.2](#)
- [Ha *et al.*, 2015] HA, Y.-U., JIN, J.-H. et LEE, M.-J. (2015). Lets3D : A Collaborative 3D Editing Tool Based On Cloud Storage. *International Journal of Multimedia and Ubiquitous Engineering*, 10(9):189–198. [2.1.1](#)
- [Hand, 1997] HAND, C. (1997). A Survey of 3D Interaction Techniques. *Computer Graphics Forum*, 16(5):269–281. [1.1.2](#)

- [Haque *et al.*, 2016] HAQUE, S. A., ISLAM, S., ISLAM, M. J. et GRÉGOIRE, J. C. (2016). An architecture for client virtualization : A case study. *Computer Networks*, 100:75–89. [2.3.2](#)
- [Hinze *et al.*, 2009] HINZE, A., SACHS, K. et BUCHMANN, A. (2009). Event-based Applications and Enabling Technologies. *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, pages 1 :1–1 :15. [1.1.3](#), [3.2](#)
- [Houston *et al.*, 2013] HOUSTON, B., CHEN, R., MCKENNA, T., LARSEN, W., LARSEN, B., CARON, J., NIKFETRAT, N., LEUNG, C., SILVER, J., KAMAL-AL-DEEN, H. et CALLAGHAN, P. (2013). Clara.io. *ACM SIGGRAPH 2013 Studio Talks on - SIGGRAPH '13*, pages 1–1. [2.1.2](#), [2.2](#)
- [Hu et Chen, 2017] HU, Y. et CHEN, Z. (2017). WebTorrent Based Fine-grained P2P Transmission of Large- scale WebVR Indoor Scenes. [2.3.2](#)
- [Jung *et al.*, 2012] JUNG, Y., BEHR, J., DREVENSEK, T. et WAGNER, S. (2012). Declarative 3D approaches for distributed web-based scientific visualization services. *CEUR Workshop Proceedings*, 869. [2.1.3](#)
- [Khronos, 2007] KHRONOS (2007). OpenGL ES 2. [2.1.3](#)
- [Khronos, 2008] KHRONOS (2008). OpenGL ES 3. [2.1.3](#)
- [Khronos, 2011] KHRONOS (2011). WebGL 1.0. [2.1.3](#)
- [Khronos, 2016] KHRONOS (2016). WebGL 2.0. [2.1.3](#)
- [Klamer, 2013] KLAMER, J. (2013). *Conflict resolution in an event sourcing environment*. Thèse de doctorat. [2.3.5](#)
- [Koskela *et al.*, 2015] KOSKELA, T., HEIKKINEN, A., HARJULA, E., LEVANTO, M. et YLIANTTILA, M. (2015). RADE : Resource-aware Distributed Browser-to- browser 3D Graphics Delivery in the Web. *IEEE Wireless and mobile*, pages 500–508. [3.2](#)
- [Koskela *et al.*, 2014] KOSKELA, T., VATJUS-ANTTILA, J. et DAHL, T. (2014). Communication Architecture for a P2P-enhanced Virtual Environment Client in a Web Browser. pages 1–5. [2.2.4](#)
- [Kosmadoudi *et al.*, 2013] KOSMADOUDI, Z., LIM, T., RITCHIE, J., LOUCHAR, S., LIU, Y. et SUNG, R. (2013). Engineering design using game-enhanced CAD : The potential to augment the user experience with game elements. *CAD Computer Aided Design*, 45(3). [1.1.2](#)
- [Kounev *et al.*, 2008] KOUNEV, S., BACON, J., SACHS, K. et BUCHMANN, A. (2008). A methodology for performance modeling of distributed event-based systems.

- Proceedings - 11th IEEE Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, ISORC 2008*, pages 13–22. [2.3.2](#)
- [Lewis, 1995] LEWIS, J. (1995). IBM Computer Usability Satisfaction Questionnaires : Psychometric Evaluation and Instructions for Use. *International Journal of Human-Computer Interaction*, 7(1):57–78. [5.4](#)
- [Lewis, 1993] LEWIS, J. R. (1993). Multipoint scales : Mean and median differences and observed significance levels. *International Journal of Human-Computer Interaction*, 5(4):383–392. [5.4](#)
- [Li et al., 2015] LI, J., CHOU, J.-K. et MA, K.-L. (2015). High performance heterogeneous computing for collaborative visual analysis. *SIGGRAPH Asia 2015 Visualization in High Performance Computing on - SA '15*, pages 1–4. [2.3.2](#), [2.3.2](#)
- [Lowet et Goergen, 2009] LOWET, D. et GOERGEN, D. (2009). Co-Browsing Dynamic Web Pages. *Proceedings of the 18th International Conference on World Wide Web - WWW '09*, pages 941–950. [2.1.3](#)
- [Lu et al., 2016] LU, Z., GUERRERO, P., MITRA, N. J. et STEED, A. (2016). Open3D : Crowd-Sourced Distributed Curation of City Models. *Web3D '16 : Proceedings of the 21th International Conference on 3D Web Technology*, pages 87–94. [2.1.2](#)
- [Martinez G. et al., 2009] MARTINEZ G., A., OROZCO, H., RAMOS, F. et SILLER, M. (2009). A Peer-to-Peer Architecture for Real-Time Distributed Visualization of 3D Collaborative Virtual Environments. *2009 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*. [2.2.4](#)
- [Montresor et Jelasity, 2009] MONTRESOR, A. et JELASITY, M. (2009). PeerSim : A Scalable P2P Simulator. *IEEE P2P'09 - 9th International Conference on Peer-to-Peer Computing*, (214412):99–100. [2.3.2](#)
- [Morin, 1990a] MORIN, E. (1990a). *Introduction à la pensée complexe*. ESF Editeur. [1](#)
- [Morin, 1990b] MORIN, E. (1990b). Science avec conscience. [1.1.1](#)
- [Mouton et al., 2011] MOUTON, C., GRIMSTEAD, I. et CARDIFF, U. (2011). Collaborative Visualization Current Systems and Future Trends. *Proceedings of the 16th International Conference on 3D Web Technology*, 1:101–110. [2.1.2](#)
- [Mouton et al., 2014] MOUTON, C., PARFOURU, S., JEULIN, C., DUTERTRE, C., GOBLET, J.-L., PAVIOT, T., LAMOURI, S., LIMPER, M., STEIN, C., BEHR, J. et JUNG, Y. (2014). Enhancing the Plant Layout Design Process using X3DOM and a Scalable Web3D Service Architecture. [2.2](#)

- [Ogden *et al.*, 2017] OGDEN, M., MCKELVEY, K. et MADSEN, M. B. (2017). Dat -Distributed Dataset Synchronization And Versioning. (May). [2.2.4](#)
- [Oki *et al.*, 1993] OKI, B., PFLUEGL, M., SIEGEL, A. et SKEEN, D. (1993). The Information Bus. *Proceedings of the fourteenth ACM symposium on Operating systems principles - SOSP '93*, pages 58–68. [2.3.2](#)
- [Pacull *et al.*, 1994] PACULL, F., SANDOZ, A. et SCHIPER, A. (1994). Duplex : A Distributed Collaborative Editing Environment in Large Scale. *Proceedings of the 1994 ACM conference on Computer supported cooperative work - CSCW '94*, pages 165–173. [2.1.1](#)
- [Papageorgiou *et al.*, 2011] PAPAGEORGIOU, N., VERGINADIS, Y., APOSTOLOU, D. et MENTZAS, G. (2011). Collaboration pattern assistant. *Proceedings of the 5th ACM international conference on Distributed event-based system - DEBS '11*, page 387. [2.3.1](#)
- [Parzyjegla *et al.*, 2010] PARZYJEGLA, H., GRAFF, D., SCHRÖTER, A., RICHLING, J. et MÜHL, G. (2010). Design and implementation of the Rebeca publish/subscribe middleware. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6462 LNCS, pages 124–140. Springer, Berlin, Heidelberg. [2.3.2](#)
- [Pietzuch et Bacon, 2002] PIETZUCH, P. et BACON, J. (2002). Hermes : a distributed event-based middleware architecture. *22nd International Conference on Distributed Computing Systems Workshops (ICDCS 2002)*, pages 611–618. [2.3.2](#)
- [Prakash et Knister, 1994] PRAKASH, A. et KNISTER, M. J. (1994). A framework for undoing actions in collaborative systems. *ACM Transactions on Computer-Human Interaction*, 1(4):295–330. [2.1.1](#)
- [Riley et Henderson, 2010] RILEY, G. F. et HENDERSON, T. R. (2010). The ns-3 Network Simulator. In *Modeling and Tools for Network Simulation*, pages 15–34. Springer Berlin Heidelberg, Berlin, Heidelberg. [2.3.2](#)
- [Roberto *et al.*, 2014] ROBERTO, D., DIAS, C., DURELLI, R. S., REMO, J., BREGA, F., GNECCO, B. B., TREVELIN, L. C. et GUIMARÃES, M. D. P. (2014). Data Network in Development of 3D Collaborative Virtual Environments : A Systematic Review. *LNCS*, 8579:769–785. [2.2.4](#)
- [Shapiro et Preguiça, 2007] SHAPIRO, M. et PREGUIÇA, N. (2007). Designing a commutative replicated data type. *arXiv preprint arXiv :0710.1784*. [2.1.1](#)
- [Singhal et Zyda, 1999] SINGHAL, S. et ZYDA, M. (1999). Networked Virtual Environments. (April):222–226. [1.1.2](#)

- [Sons *et al.*, 2010] SONS, K., KLEIN, F., RUBINSTEIN, D., BYELOZYOROV, S. et SLUSALLEK, P. (2010). XML3D – Interactive 3D Graphics for the Web. *Proceedings of the 15th International Conference on Web 3D Technology*, pages 175–184. [2.1.3](#)
- [Steiakaki *et al.*, 2016] STEIAKAKI, M., KONTAKIS, K. et MALAMOS, A. G. (2016). Real-Time Collaborative environment for interior design based on Semantics , Web3D and WebRTC. *International Symposium on Ambient Intelligence and Embedded Systems*, pages 22–24. [3.3](#)
- [Stein *et al.*, 2014] STEIN, C., LIMPER, M. et KUIJPER, A. (2014). Spatial data structures for accelerated 3D visibility computation to enable large model visualization on the web. *Proceedings of the Nineteenth International ACM Conference on 3D Web Technologies - Web3D '14*, pages 53–61. [2.1.3](#)
- [Steinfeld *et al.*, 1999] STEINFELD, C., JANG, C.-Y. et PFAFF, B. (1999). Supporting Virtual Team Collaboration - TeamSCOPE System. *International Conference on Supporting Group Work (GROUP'99)*, pages 81–90. [1.1.3](#)
- [Sun, 2002] SUN, C. (2002). Undo as concurrent inverse in group editors. *ACM Transactions on Computer-Human Interaction*, 9(4):309–361. [2.1.1](#)
- [Sun *et al.*, 1998] SUN, C., JIA, X., ZHANG, Y., YANG, Y. et CHEN, D. (1998). Achieving Convergence , Causality Preservation , and Intention Preservation in Real-Time Cooperative Editing Systems. *ACM Trans. Comput.-Hum. Interact.*, 5(1):63–108. [3.3](#)
- [Sung *et al.*, 2006] SUNG, M. Y., YOO, Y., JUN, K., KIM, N. J. et CHAE, J. (2006). Experiments for a collaborative haptic virtual reality. *Proceedings - 16th International Conference on Artificial Reality and Telexistence - Workshops, ICAT 2006*, pages 174–179. [2.2.4](#)
- [Sutter, 2015] SUTTER, J. (2015). A CSS Integration Model for Declarative 3D. *In Web3D '15 : Proceedings of the 20th International Conference on 3D Web Technology*, pages 209–217. [2.1.3](#)
- [Tarkoma, 2012] TARKOMA, S. (2012). Research Solutions. *In Publish/Subscribe Systems*, pages 205–237. John Wiley & Sons, Ltd, Chichester, UK. [2.3.2](#)
- [Verginadis *et al.*, 2009] VERGINADIS, Y., APOSTOLOU, D., PAPAGEORGIOU, N. et MENTZAS, G. (2009). Collaboration Patterns in Event-Driven Environments for Virtual Organizations. Rapport technique. [2.3.1](#)
- [Vernon, 2013] VERNON, V. (2013). *Implementing Domain-Driven Design*. Addison-Wesley Longman Publishing Co., Inc. [2.3.3](#)

- [W3C, 2011] W3C (2011). Extensible 3d (X3D). [2.1.3](#)
- [Weiss *et al.*, 2009] WEISS, S., URSO, P. et MOLLI, P. (2009). An Undo Framework for P2P Collaborative Editing. In BERTINO, E. et JOSHI, J. B. D., éditeurs : *Collaborative Computing : Networking, Applications and Worksharing : 4th International Conference, CollaborateCom 2008, Orlando, FL, USA, November 13-16, 2008, Revised Selected Papers*, pages 529–544. [2.1.1](#)
- [Wu *et al.*, 2014] WU, D., ROSEN, D. W. et SCHAEFER, D. (2014). *Cloud-Based Design and Manufacturing (CBDM)*. [1.1](#)
- [Xhafa et Poulouvassilis, 2010] XHAFA, F. et POULOVASSILIS, A. (2010). Requirements for distributed event-based awareness in P2P groupware systems. *24th IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2010*, (October):220–225. [1.1.3](#), [3.2.1](#)
- [You et Pekkola, 2001] YOU, Y. et PEKKOLA, S. (2001). Meeting others—supporting situation awareness on the WWW. *Decision Support Systems*, 32:71–82. [1.1.3](#)
- [Young, 2009] YOUNG, G. (2009). Code Better. [2.3.4](#)
- [Yu *et al.*, 2016] YU, M., CAI, H., MA, X. et JIANG, L. (2016). Symmetry-Based Conflict Detection and Resolution Method towards Web3D-based Collaborative Design. *Symmetry*, 8(5):35. ([document](#)), [2.1](#)
- [Yuan *et al.*, 2002] YUAN, P.-p. Y. P.-p., CHEN, G. C. G., DONG, J.-x. D. J.-x. et HAN, W.-l. H. W.-l. (2002). Research on an event specification for event-based collaboration support software architecture. *The 7th International Conference on Computer Supported Cooperative Work in Design*, 7:99–104. [2.3.1](#)
- [Zhu *et al.*, 2011] ZHU, M., MONDET, S., MORIN, G., OOI, W. T. et CHENG, W. (2011). Towards peer-assisted rendering in networked virtual environments. *Proceedings of the 19th ACM international conference on Multimedia - MM '11*, page 183. [2.2.4](#)