3DEvent: A Framework Using Event-Sourcing Approach For 3D Web-Based Collaborative Design in P2P

Caroline Desprat[*] University of Toulouse

Jean-Pierre Jessel[†] University of Toulouse

Hervé Luga[‡] University of Toulouse

Abstract

Despite recent advances, especially in web-based Collaborative Virtual Environments (CVEs) using real-time 3D content, Web technology still requires an efficient way to distribute and stream large-scale 3D data. In this paper, we present 3DEvent: an event-driven framework to collaboratively manipulate predesigned 3D content in real-time on a web-based platform. This work introduces a new approach in achieving 3D object manipulation tasks during collaborative design stages using event-sourcing. Usually, a client-server architecture supports updates to the 3D environment state. Peer-to-peer (P2P) allows direct communication between teammates reducing response times during collaboration and decreasing server load, reducing the costs of providers. 3DEvent enables P2P-assisted delivery of 3D dynamic content in a web browser via WebRTC. By combining concepts from distributed event-processing and mesh-processing, 3D independent rendering and event-based synchronization, we present 3DEvent framework and potential uses associated that support history-aware 3D applications into a unified distributed processing solution for 3D web-based CVEs.

Keywords: collaborative manipulation, distributed application, event-driven architecture, Web 3D, WebRTC.

Concepts: •**Human-centered computing** → **Web-based interaction;** *Asynchronous editors;* •**Software and its engineering** → **Publish-subscribe / event-based architectures;** •**Computer systems organization** → *Distributed architectures;*

1 Introduction

In recent years, many technology companies have transferred a significant portion of their software solutions to the web. This is the result of the increasing need for mobility of collaborators and the geographic distance between them. A good example of this trend is the market's development of web-based solutions for 3D content editing (Clara.io [Houston et al. 2013], GrabCAD, Verold Studio). The increasing size and complexity of datasets (such as CAD data) make them more difficult to transmit over the Internet and render in web browsers. In a collaborative domain, a lot of concurrent users are involved; data update conflicts are more likely to occur because the update operations take place on a single datum. Unless there is an additional auditing mechanism, which records the details of each operation in a separate log, the history is lost. In industrial and learning context, data manipulations (import, transformations...) need history-tracking feature to prevent losing information. History-awareness concept embeds not only undo/redo commands (short-term history) but also long-term versioning system to offer real-time monitoring and past collaborations reviewing to better understand the construction of the content in a scene. The Collaborative Virtual Environment (CVE) needs to be robust, scalable, and consistent for all clients. These conditions can be hard to handle with traditional methods (client-server or saving the complete object's state after modification...). Consequently, remote collaboration in interactive environments have to be adapted to handle size complexity of 3D content (reactivity, update-on-demand) and alleviate the load on the server. While classic client-server communication can easily manage dozens of client editing in a virtual world, servers can become a bottleneck (massive data generation of 3D collaboration, crashes). Also, the development of the area of event-processing [Chandy et al. 2011] provides relevant opportunities with real or near-real time reactivity for ressource management. Coupling the client-server architecture with a P2P network should facilitate collaboration between users, keep the environment consistent and detect collaboration conflicts, providing robustness by distributing and replicating relevant data horizontally. In this work, we consider high bandwidth and reliable network. The types of data processed are 3D geometry meshes without texture.

Contribution We propose an hybrid communication architecture for web-based CVE applications. Our framework provides a strong history-aware usability through P2P-assisted 3D dynamic content delivery in the form of events in a lightweight environment to alleviate the load on server during collaboration. Event-sourcing pattern used in 3DEvent permits an homogeneous data lifecycle management allowing pertinent collaborative features such as semantics, versioning, and conflict detection. By exploiting exclusively client resources for networking, 3D data processing, and visualization, we ensure user's autonomy in collaborative 3D content creation by deporting business expertise in the browser (offline working) and provide robustness in case of server issues.

This paper is organized as follows: Section 2 considers related works in web-based CVEs and event-based (EB) architectures. Section 3 presents the 3DEvent framework and a co-design application as illustration. Section 4 discusses our methodology and current limitations while Section 5 concludes the paper with ongoing works optimizations for larger contents.

2 Related works

Web-based collaborative environments With the rise of HTML5 and more powerful clients, pluginless solutions are now well supported on the web (WebGL, X3D). Many academic works [Brown et al. 2003][Grasberger et al. 2013][Mouton et al. 2014] and commercial solutions (Onshape, GrabCAD) using elaborate versioning systems for web-based CVEs have been proposed. However, these popular systems were client-server based and relied on full transfer of large 3D data for each client. This architecture can provoke a bottleneck at the server (or required flexible server-side scalability that can be expensive) and were not usable offline. To al-

[*]e-mail:desprat@irit.fr
[†]e-mail:jessel@irit.fr
[‡]e-mail:luga@irit.fr

DOI: http://dx.doi.org/10.1145/2945292.2945310

leviate the server load, low cost solutions using P2P communication with WebRTC [Jennings et al. 2014] have emerged to decentralize distribution for large amount of data [Zhang et al. 2013]. [Li et al. 2015] and [Koskela et al. 2015] have shown that crowd computing using hybrid architecture benefit collaborative visual analysis or 3D static asset delivery but the server's role was still prominent. [Desprat et al. 2015] proposed dynamic content distribution system with a costly data transmission and no history/versioning features.

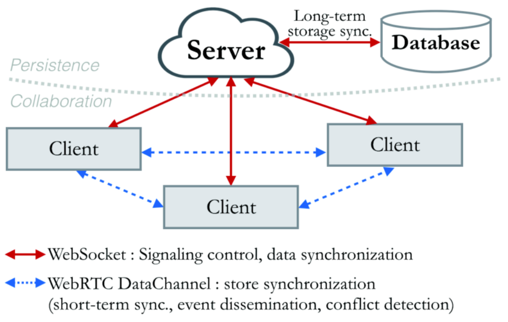
Event-driven architectures for collaboration Conventional applications alter states by replacing values. The effective state only shows the latest version of the application and omits previous modifications. Moreover, states are overwritten and lost at each change. During conception in CAD, the history is equally important as the 3D model itself. The benefits of EB architectures have been proven in distributed architectures with loose-coupled and non-monolithic design, openness and scalability properties [Hohpe 2006]. Event-sourcing approach consists of ensuring the capture of every state's change of an application in an event object [Fowler 2003]. These event objects are stored in sequence for the same lifetime as the application state itself. The application uses an append-only store to record it in an ordered event log instead of updating states. This event log allows the system to be aware of the current state of an object but also of its entire history. Each event is autonomous (holds all it needs) and has semantics related to the expert domain. For instance, CoDesign [Bang et al. 2010] was an EB conflict detection framework for collaborative modeling applying rules (given/when/then) on the server-side. In this centralized approach the creation of expert content in offline situations is not possible. [Xhafa and Poulovassilis 2010] showed the benefits of decentralized collaborative systems for EB awareness in groupware systems.

3 3DEvent Framework

In this section, we will describe the parts of our framework: the communication architecture, the client architecture, the network synchronization between clients, and persistences (short-term and long-term). The data distribution is event-based, meaning that each action done is translated into domain event instances and sent into a packet through the network. Events are used to instantiate 3D domain-related objects, send information for updating objects and provide other meta-information (conflict detection, triggered process…). The content distribution cinematic in 3DEvent is shared between two levels: network communication architecture and client architecture. We will detail each bridge components, from users actions to 3D rendering. Then, synchronization mechanisms between peers and long-term persistence will be explained. Finally, we will present a co-design application example to validate our framework.

3.1 Hybrid communication architecture

The communication architecture for adaptive 3D entities is illustrated in Figure 1. This full-web hybrid architecture was previously proposed in [Desprat et al. 2015]. Here, we revise their client architecture to integrate event-sourcing in the collaboration layer. Components of our communication architecture are clients, server and database. Each client should implement the WebGL standard, the WebSocket protocol for client-server communication (long-term persistence), the WebRTC DataChannel [Grigorik 2013] protocol for asynchronous synchronization, and a short-term persistence storage. The server handles the communication with long-term persistence storage (database) synchronized, with content creation when clients are online, and queried when a user needs a content that is not present in the collaboration network. The communication between clients and server uses WebSocket protocol.



Server
Long-term storage sync.
Database
Persistence
Collaboration
Client
Client
Client
WebSocket : Signaling control, data synchronization
WebRTC DataChannel : store synchronization (short-term sync., event dissemination, conflict detection)

Figure 1: *Hybrid client-server and P2P architecture*

3.2 Client platform architecture

The main part of the framework is represented in both Figure 2 and Figure 3: event's representation and the client architecture. To avoid event-sourcing implementation over client-server architecture, we embed it only on the client side to have offline access to the expertise in the browser. As in P2P a peer is both client and server, this approach is adapted to counter sudden disconnection, server crash.… We investigated the description of an event [Tominski 2006] illustrated in Figure 2 and adapted its properties for 3D content. *Event domain* (*ED*) is the bounded context in which events occur. It contains entities with respect of specified *event type* (*ET*). An *ET* is used to express a concrete interest regarding the entities of an *ED* –*abstract events* (*aet*) are *ET* compatible with *ED*–. *Event instances* (*EI*) (or short *event*) is composed of the triple *ED*,*ET*, and *EP*. *EP* denotes *event parameters* assigned to an *EI*. Event specification requires compiling several types of events that are interesting for manipulation tasks. Event detection retrieves an instance of specified event types to be visualized. In this work, we evaluate conditions to trigger the appropriate event. Event representation is the reflection on the application of the triggered events behavior.

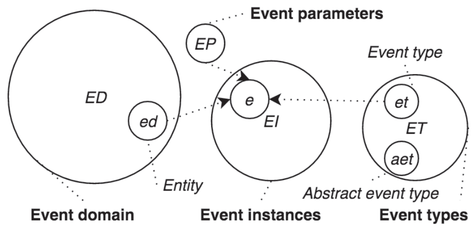


Figure 2: *Illustration of the notions event domain, event type, and event instance.*

In our system, the domain is the representation of 3D objects from an expert's point of view. The domain objects represents 3D data in an abstracted format (geometry, position), independent of rendering needs (lighting, material...). 3DEvent integrates an event generator: a programming component designed to provide allow the creation of event types adapted to 3D user's interests (semantics).

In the schema of Figure 3, we represented the data's lifecycle in our framework inside a web-browser. User actions are issued from the task-based UI described in Section 3.4. When the user does actions, the domain validates their parameters and generates new version of aggregates. An aggregate is an encapsulation that bounds a transaction. It handles commands, applies events and has a state model encapsulated within it. Thus, the aggregate can apply the required validation of command and uphold its invariants (business rules). In our system, aggregates are have Scene, Mesh, and Geometry.

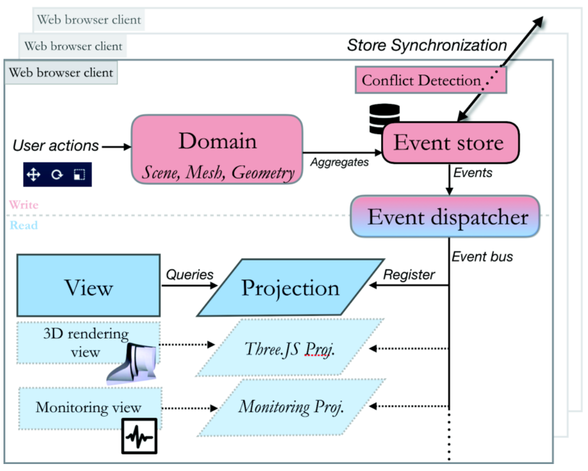


Figure 3: *Client architecture model: data lifecycle management with event-sourcing in a web-browser from user's actions to visualization through network synchronization.*

The event store records the events according to the aggregates processed and external events received from the network (short-term persistence). It synchronizes itself with other clients and the server database (long-term persistence). Then events are sent to the event dispatcher that delivers asynchronously events according to the registered services on it. We propose two projection services: 3D rendering with Three.JS and monitoring. A projection derives from a stream of events to provide (filter, enrich) adapted objects for the view. The rendering pipeline is responsible for the re-ordering of the received events from the event bus: if the Mesh event arrives before the creation of the Geometry, it is pushed on the waiting list until the needed events come. Therefore, the out-of-order nature of WebRTC and asynchronous communication is no longer an issue.

3.3 Store synchronization during collaboration

The P2P network is a partial mesh topology graph; the peer connectivity policy is defined by the server and adaptable according to the needs. The P2P mesh is automatically built when the client is connecting to the server. The server provides Ids to each client with the Ids of collaborator to connect with. Then, the client initiates the P2P connection. Once connected, collaborators can exchange data via a bidirectional data connections (WebRTC DataChannel) avoiding passing through the server again. In case of crash and recovery of the server, the collaboration can continue. Each event created by a user's command is pushed into a message and spread to other collaborators through the channels: if the event has not been applied yet, the message is processed and spread to next P2P neighbors; otherwise, the message dies silently. Each message contains a set of events, each one containing its Id, its version and a timestamp that will be checked at the reception to order events if necessary.

The conflict detection component allows the developer to implement its own conflict resolution rules. It triggers a flag if versions between the received aggregate and the current one are equal (see Figure 4). According to the business logic rules defined, the event is rejected, accepted with or without modifications. From this process, new events can be generated during the resolution.

3.4 3D environment

The client hosts a 3D environment to visualize, upload, and manipulate data with basic transformations (translation, rotation, and

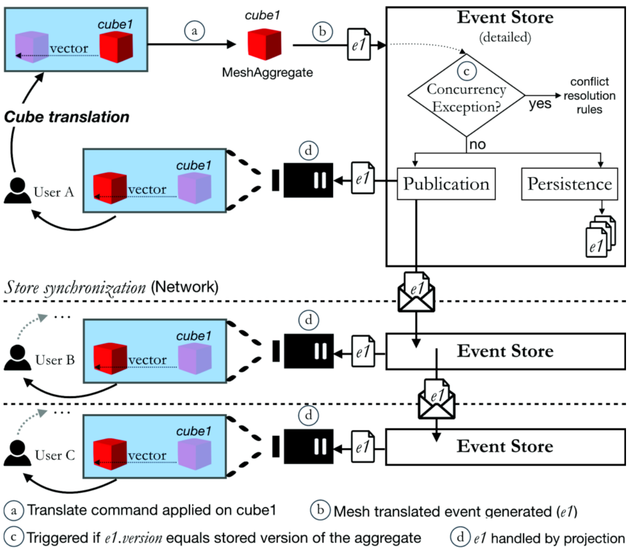

cube1
vector
a
cube1
MeshAggregate
b
e1
Event Store
(detailed)
c
Concurrency Exception?
yes
conflict resolution rules
no
Publication
Persistence
Cube translation
User A
d
Store synchronization (Network)
User B
Event Store
User C
Event Store
a Translate command applied on cube1
b Mesh translated event generated (e1)
c Triggered if e1.version equals stored version of the aggregate
d e1 handled by projection

Figure 4: *Collaborative editing example where User A is connected to User B, itself connected to User C. The cycle shows the steps from the triggered action to event generation, store synchronization and impact on users' rendering for a cube translation.*

scale) and navigation through the scene. New technologies and standards such as WebGL and Web Workers have made recent web browsers to support 3D rendering and parallel computing. Thus, the 3D mesh processing is dedicated to the GPU for a full use of the CPU capacity. As the 3D rendering projection is mandatory in 3DEvent, Three.JS library is proposed by default.

Our framework proposes first-come-first-serve (FCFS) conflict resolution model by default. If two users are trying to edit the same object, the conflict is avoided by creating a clone of the "real" object on selection. During transformation, the user manipulates the clone until deselection. Thus, the other user can simultaneously manipulate another clone of the same object without conflict. The first user to unselect the object will trigger the move the "real" object, allowing the second user to cancel or finish its own action. Each user is assigned a color to show which object is manipulated by whom, and a cone as its point of view to express its presence in the scene.

3.5 3D editor application example

To validate our approach, we designed a lightweight and cross-platform prototype using isomorphic JavaScript: a 3D editor with a geometry library. We chose to use Node.js as a runtime environment, PeerJS as P2P communication channels and ThreeJS as 3D WebGL engine. Messages contain the event data in JSON. The persistence is computed from the JSON delta diffs out of the old and new states so the 3D can be rendered to the user. The prototype uses an in-memory event store. The application provides high level 3D interactions that can be designed as an event (9 events in total) such as create/delete scene, import geometry into the library, create mesh (by dropping to a specific position or clicking on the item), delete meshes, use mesh translation, rotation or scale. Figure 4 describes how the system executes a translation action triggered by a user and how it's broadcasted to his/her collaborators (scene, cube geometry, and cube mesh should be created before). During the sessions, until five (fake) collaborators in LAN in partial mesh connectivity were set up to test the online co-design features and server or client crashes to observe the system behavior. As a result, clients can interact in easily with 3D objects in the scene and see others' modifications. They are able to recover new events after offline working or disconnection. A server crash does not impact current collaboration because new users can work offline, and session collaborators are not disconnected from each other.

4 Discussion

We have structured our methodology analyzing and evaluating related concepts and approaches of existing systems. We have defined a computational model to address our requirements and unify different concepts. Then, we have derived a framework and a corresponding environment adapted to our network constraints. This work is intended to improve the next prototype proposing an abstraction of the application model by incorporating event-sourcing for the computational and programming model. As a result, we implemented a 3D editor prototype application to show how one 3D application would work in practice. The framework provides independent, scalable, and robust components (data management, network architecture). 3DEvent framework eases implementation of expert systems using 3D CVE. Also, web-based solution allows a single implementation for a cross platform application using adaptive rendering (projections) in a lightweight environment.

Limitations Our framework is based on young technologies; WebRTC Datachannel protocol is still a draft standard thus it can only run on Chrome and Firefox latest versions. 3DEvent can handle scenes with dozens of small models (<15MB) and their versions. When come bigger models, the choice of the granularity (vertex granularity vs. geometry granularity) may introduce more processing if not adapted by the developer. Still, P2P eases the absorption of traffic compared to client-server application. Because we framed our context on small team collaboration, the conflict resolution system is currently very simple. For more complex managements, adding expert rules using the expected version of the aggregate is needed. The update or recovery of events is executed from all past events that can be costly at long term without using event-sourcing's snapshot concept when the sum of events is bigger than a state. To fulfill our validation, a quantitative evaluation of the framework should integrate throughput, latency, number of users collaborating, and model's size criteria.

5 Conclusion

3DEvent framework introduces an original event-based approach for generating 3D web-based CVEs for object manipulation. The full-web hybrid architecture allows a better transmission of updates between collaborators without overloading server's channels. Event-sourcing pattern shows several benefits for this solution needing : history, lightweight update messages, and analytics. Our framework can supply the adaptation and personalization of complex visualizations thanks to the event-based approach. Because of this flexibility, the plasticity of the 3D CVEs' interfaces will be easier to integrate later. 3DEvent is a response to the impact of geographical mobility of experts and widespread adoption of more and more performant mobile solutions that can be used in 3D collaborative visualization, manipulation and editing needs.

In the future, we will aim to enhance P2P 3D data streaming using compression for collaborative and 3D content in real network conditions. In a 3D design environment with preconceived objects allowing high-level transformations, we want to address streaming issues for larger objects containing thousands of connected components using progressive meshes. This would alleviate transmission load without degrading user experience by personalizing the projection (view). It would generates the adapted content for users in a 3D responsive environment (view-dependent and multi-resolution parts of the object) while integrating a load balancing method as shown in [Li et al. 2015] since our events are not ordered.

References

BANG, J. Y., POPESCU, D., EDWARDS, G., MEDVIDOVIC, N., KULKARNI, N., RAMA, G. M., AND PADMANABHUNI, S. 2010. CoDesign: a highly extensible collaborative software modeling framework. *2010 ACM/IEEE 32nd Int. Conf. Softw. Eng. 2*, 243–246.

BROWN, D., JULIER, S., BAILLOT, Y., AND LIVINGSTON, M. 2003. An event-based data distribution mechanism for collaborative mobile augmented reality and virtual environments. *IEEE Virtual Reality, 2003. Proceedings. 2003*.

CHANDY, M. K., ETZION, O., AND AMMON, R. V. 2011. The event processing manifesto. *Event Process.*, 10201, 1–60.

DESPRAT, C., LUGA, H., AND JESSEL, J.-P. 2015. Hybrid client-server and P2P network for web-based collaborative 3D design. *WSCG 2015 Conf. Comput. Graph. Vis. Comput. Vis.*, 229–238.

FOWLER, M. 2003. *Patterns of Enterprise Application Architecture*, vol. 23. Addison-Wesley Longman Publishing Co., Inc.

GRASBERGER, H., SHIRAZIAN, P., WYVILL, B., AND GREENBERG, S. 2013. A data-efficient collaborative modelling method using websockets and the BlobTree for over-the air networks. *Proc. 18th Int. Conf. 3D Web Technol. - Web3D '13*, 29.

GRIGORIK, I. 2013. *High Performance Browser Networking*. O'Reilly Media, Inc.

HOHPE, G. 2006. Programming Without a Call Stack Event-driven Architectures. *Enterp. Integr. Patterns*.

HOUSTON, B., CHEN, R., MCKENNA, T., LARSEN, W., LARSEN, B., CARON, J., NIKFETRAT, N., LEUNG, C., SILVER, J., KAMAL-AL-DEEN, H., AND CALLAGHAN, P. 2013. Clara.io. *ACM SIGGRAPH 2013 Stud. Talks - SIGGRAPH '13*, 1–1.

JENNINGS, C., NARAYANAN, A., BURNETT, D., AND BERGKVIST, A. 2014. WebRTC 1.0: Real-time communication between browsers. *W3C, W3C Ed. Draft. Aug*.

KOSKELA, T., HEIKKINEN, A., HARJULA, E., LEVANTO, M., AND YLIANTTILA, M. 2015. RADE : Resource-aware Distributed Browser-to- browser 3D Graphics Delivery in the Web. *IEEE Wirel. Mob.*, 500–508.

LI, J., CHOU, J.-K., AND MA, K.-L. 2015. High performance heterogeneous computing for collaborative visual analysis. *SIGGRAPH Asia 2015 Vis. High Perform. Comput. - SA '15*, 1–4.

MOUTON, C., PARFOURU, S., JEULIN, C., DUTERTRE, C., GOBLET, J.-L., PAVIOT, T., LAMOURI, S., LIMPER, M., STEIN, C., BEHR, J., AND JUNG, Y., 2014. Enhancing the Plant Layout Design Process using X3DOM and a Scalable Web3D Service Architecture.

XHAFA, F., AND POULOVASSILIS, A. 2010. Requirements for distributed event-based awareness in P2P groupware systems. *24th IEEE Int. Conf. Adv. Inf. Netw. Appl. Work. WAINA 2010*, October, 220–225.

ZHANG, L., ZHOU, F., MISLOVE, A., AND SUNDARAM, R. 2013. Maygh: Building a CDN from client web browsers. *Proc. 8th ACM Eur. Conf. Comput. Syst. EuroSys 2013*, 281–294.