SYMPOSIUM ON SIMULATION IN ARCHITECTURE + URBAN DESIGN
SimAUD 2020

SpeckleViz: A Web-based Interactive Activity Network Diagram for AEC

Paul Poinet[1], Dimitrie Stefanescu[1], Eleni Papadonikolaki[1]

[1]University College London
Bartlett School of Construction and Project Management
London, United Kingdom
{p.poinet, d.stefanescu, e.papadonikolaki}@ucl.ac.uk

ABSTRACT

As architectural design and construction projects tend to tackle larger scales and become more complex, the multiple involved disciplines in the Architecture, Engineering and Construction (AEC) sector often need to work globally from different remote locations. This increased complexity impacts digital design up until to fabrication workflows, which become more challenging and discontinuous, as each industry partner involved in the construction of a given project operates on different software environments and needs to access the precise fabrication data of specific design components. Consequently, managing and keeping track of design changes and data flow throughout the whole design process still remains a challenging task. This paper discusses how this particular challenge can be tackled through the development of a web-based interactive Activity Network Diagram (AND) - named SpeckleViz - that continuously maps the data transfers of the design and building processes, enabling the end-user to explore, interact and get a better understanding of the constantly evolving digital design workflow. Through this paper, the authors qualify an "end-user" as an advanced or expert user that performs complex geometry modelling tasks within wider collaborative workflows involving other advanced end-users. SpeckleViz (2020) is an application built upon Speckle (2020), an open-source data platform for the AEC. We illustrate the usefulness of interactive visualization of ANDs in the development of digital design workflows.

Author Keywords
Digital Workflow; Project Management; Data Visualization; User Interface; Activity Network Diagram.

ACM Classification Keywords
Project management techniques; Visualization techniques; Graph theory; User interface design; User centred design; Activity centred design.

1 INTRODUCTION

Design is a key phase across projects' lifecycle and collaboration among design actors is complex and crucial for project success. A recent report released by the Association of Project Management (APM) emphasizes the need for developing custom specific solutions to tackle contemporary large-scale and complex projects (Davies 2019). Indeed, contemporary design to manufacture process of large-scale and geometrically complex architectural projects remains a significant challenge, even though digital literacy keeps improving and computational design knowledge becomes more available.

It is not enough to be able to model complex geometry, but the design process must also be curated, shared and understood in more simple, transparent and intuitive ways than it is currently taking place within the Architecture, Engineering and Construction (AEC) industry. In this regard, previous research has demonstrated that simplifying intricate workflows through custom interactive visualization features enabled the end-users to better understand complex processes taking place across multiple stakeholders or software engineers.

In the AEC realm, current design processes are still segregated, and laborious manual interventions sometimes become a daily routine. In order to tackle this issue, custom management and visualization tools enabling better understanding and curation of complex projects activity networks have been proposed by different firms and individuals. Those proposed solutions converge towards the need for defining low level open-source infrastructures enabling more transparent collaborative workflows.

The present paper reviews both these contemporary solutions, the web-based open-source interoperability framework Speckle (described in section 3) that starts to gain traction in the built environment, and SpeckleViz: an interactive activity network diagram built within Speckle.

This research paper is divided into six sections: the present introduction, a state of the art in managing and visualizing complex digital design workflows (both in practice and through existing standards), an introduction to the Speckle framework, a description of the developed SpeckleViz interface, an illustration of the SpeckleViz interface through the description a selected case study, a discussion section and final concluding remarks with an outline of future works.

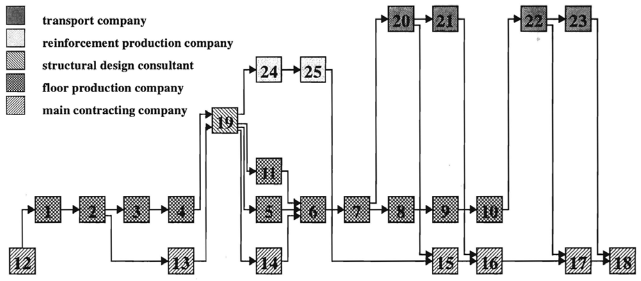

Figure 1. The Activity Network modelled as a PERT (Program Evaluation and Review Technique) network by de Vries (1995).

2 RELATED WORKS IN VISUALISING DESIGN PROCESSES

2.1 De Vries' Activity Network

Bauke de Vries (1995) conceptualized an Activity Network for the AEC industries in "Message Development in the Building Process". The author defined the activity network as follows: *"An activity network shows the information flow between activities. [...] The most important property of the network is that an activity can only start executing if all input channels contain the required information. The squares in figure 1 symbolize the activities. [...] The arrows symbolize the channels through which the information flows from one activity to another."* (de Vries, 1995). Such Activity Network can also be analogically compared with a Social Network, which has been defined by Wasserman and Faust (1994) as a "social structure" of actors (nodes) connected by one or more relations (ties), such as friendship or alliance. In the case of an Activity Network, the nodes represent activities and the ties represent data transfers – input(s)/output(s) – between the activities. Although more than 20 years old, De Vries' Activity Network for the AEC industries anticipated future design workflows and strategies that have been deployed since within practice. Those are illustrated within the next sections.

2.2 Front's Building Information Generation

During the realization of the City of Dreams Casino Hotel in Macau conceived by Zaha Hadid Architects, the consultancy practice Front developed a modelling strategy labelled "Building Information Generation" (Van der Heijden et al., 2015) enabling parallel generation of information and attributes necessary for further fabrication. In order to manage attributes and assign user data to the processed geometrical objects, Front developed an in-house a custom Rhino3D plug-in called "Elefront". The whole modelling process consisted of a strategic alternation between the generation of objects in Grasshopper and their subsequent storage and classification within staged Rhino3D models, within which the geometry is "frozen", thus devoid of any geometrical linkage. From these static, fixed geometrical objects were generated further information through a next iteration of Grasshopper sessions in which parametric linkage was kept. This process repeated itself until the last level of detail was modelled and ready for manufacture, fabrication and assembly (see Figure 2). In the words of the

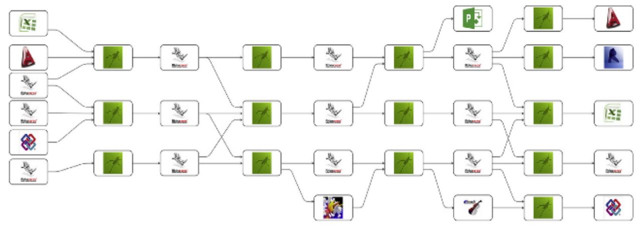

Figure 2. Project ecosystem of staged models and generating logic, across Rhino3D and Grasshopper files.

Computational Design Specialist at Front, this process is called "staging", consisting of a "discretization of logics", where many different smaller files are individually processed, instead of embedding intelligence within one single very large model. Each smaller file can be manually triggered and re-computed when a change occurs up-front, allowing therefore data propagation throughout the whole digital chain. Similarly, the developed SpeckleViz interface enable the end-user to obtain a clear overview on such discretization of logics that is operated within Speckle.

2.3 Woods Bagot's Metagraph

The architecture firm Woods Bagot has been developing internal methods and customized workflows for improving communication and software interoperability. Software platforms are not seen here as limiting the design possibilities but rather as an array of tools from which the architect can pick and choose, serving the project's needs to be designed and delivered.

Based on this approach, Woods Bagot developed Metagraph (see Figure 3), a data visualization tool based on Flux (a former interoperability platform described in the next section) data keys (or data transfer identifiers) which are used to represent all relationships between the different

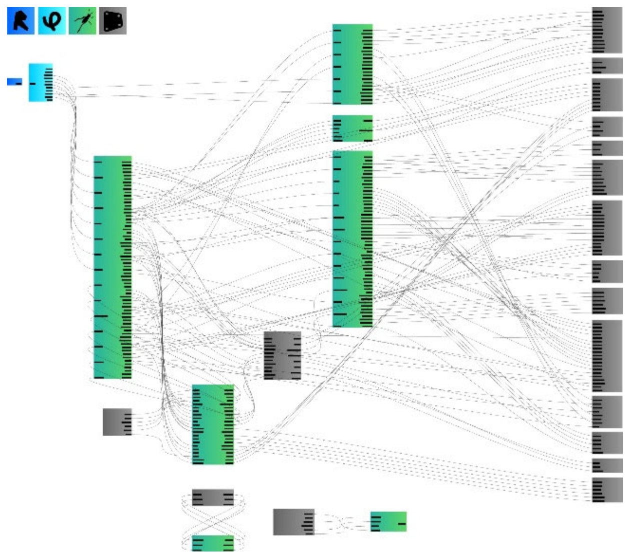

Figure 3. The Metagraph developed by Woods Bagot represents data key relationships among different scripts from different software platforms, using the Grasshopper canvas with its parameters and wire connections. (Ringley, 2017)

models, software and programming environments (Rhino, Grasshopper, Revit and Dynamo) constituting the same architectural project. The data exchange between these different software platforms are represented through a decentralized global network of nodes. While being useful for understanding and debugging models at systemic level, it is still being developed to allow for local-level programmatic control. Brian Ringley, former associate and global specialist at the design technology department of Woods Bagot, speculated that the next step of the Metagraph would be to *“set another layer of parametric intelligence where different scripts can adapt and update based on changes of other scripts.”* (Ringley, 2017)

2.4 Flux Data Inc.

Flux Data Inc. was an interoperability platform which paved the way to transfer seamlessly building data across different software platforms. It became popular within the AEC community and was largely used by many architects, engineers, and consultants before it ceased software development on the 31st of March 2018. This event was unfortunate for a substantial part of the industry, especially for those who have built up their digital workflows upon this platform. Flux had four main management tools that were accessible from the site: “Community” enabled the users to help each other on an exchange platform, “Data Explorer” helped in assessing the project's workload, “Flow” allowed to visualize the different data flows within the same project, and “Projects” enabled the different disciplines to keep track and see the current status of the projects in progress.

Depending on the client’s need, the appropriate applications could be downloaded and installed from the main website, enabling better communication between the specific software environments used by the office. If the company used a software platform that was not supported by the Flux applications, it was still possible to undertake third party software development through the available Software Development Kit (SDK). Flux software was being used by *“more than 6,200 companies in 151 countries and relied upon by Computational Designers, Engineers and sophisticated BIM professionals at Frank Gehry Partners, BIG, SHOP, Arup, BuroHappold Engineering, Thornton Tomasetti and more.”* (Flux, 2018).

2.5 Existing Standards in Process Modelling

Paralelly to the practices described above, different standards have been developed by the industry to standardise the formal representation of exchanges and activities hapenning during a business process. For example, the Business Process Modeling Notation (BPMN) has been introduced in 2004 by Stephen A. White, in order *“to provide a notation that is readily understandable by all business users, from the business analysts who create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and, finally, to the business people who will manage and monitor those processes.”* (White, 2004). The author has identify for different categories of elements: Flow Objects (Event, Activity, Gateway), Connecting Objects (Sequence Flow, Message Flow, Association), Swimlanes (Pool and Lane) and Artifacts (Data Object, Group, Annotation).

While the BPMN notation helps all business users to understand business processes more clearly, it reasonates very strongly with the Unified Modelling Language (UML), which focuses instead on modeling software system and acts as general-purpose visual modeling language intedend to specify, visualize and construct the artifacts of a software system. UML was developed in 1994-95 and was originally motivated by the desire to standardize the different notational systems and approaches to software design. UML also comes with its sets of graphical rules which helps in documenting a system model (Booch et al., 2005).

Not specially focused on business or software modelling processes but looking instead at timed event systems in a more general way, the Discrete Event System Specification (DEVS) has been introduced by Bernard P. Zeigler in 1976 to formalise the modeling and analysis of discrete event systems (DESs). A DES can be represented by a step function, and is defined as a non-linear process in which different events can happen in parallel, one event triggering an other in an asynchronous manner. This differs a lot from the more traditional, continuous imulations which could be represented by a continuous function in which optimization tasks are running one after the other (Ziegler et al., 2000).

As building design processes are not linear or continuous but discrete and intricate (as described in the previous subsections), the formalisms introduced by BPMN, UML and DEVS represent useful sources of inspiration to visualize Activities Networks in AEC. The next section introduces Speckle (2020), an open-source data platform for the AEC which enables the deployment of SpeckleViz, the Activity Network Diagram described in section 4.

3 SPECKLE, AN OPEN-SOURCE DATA PLATFORM FOR AEC

Speckle (2020) differentiates itself from commercial web-based interoperability platforms by proposing a complete open-source data framework for architects, designers and engineers. Speckle was originally developed at University College London in 2016 by Dimitrie Stefanescu. Speckle does not enforce a predefined topology of communication patterns, but rather allows for the emergence (and analysis) of meaningful data-driven dialogue amongst the different actors involved in the design process

With regards to schemas, Speckle, in contrast with the existing industry standard IFC (Industry Foundation Classes), promotes composability over completeness and provides a programmatic infrastructure for end-users to define their own, domain-, company-, or even project-specific, object models. Furthermore, Speckle can support pre-existing object models (such as IFC) “out of the box”,

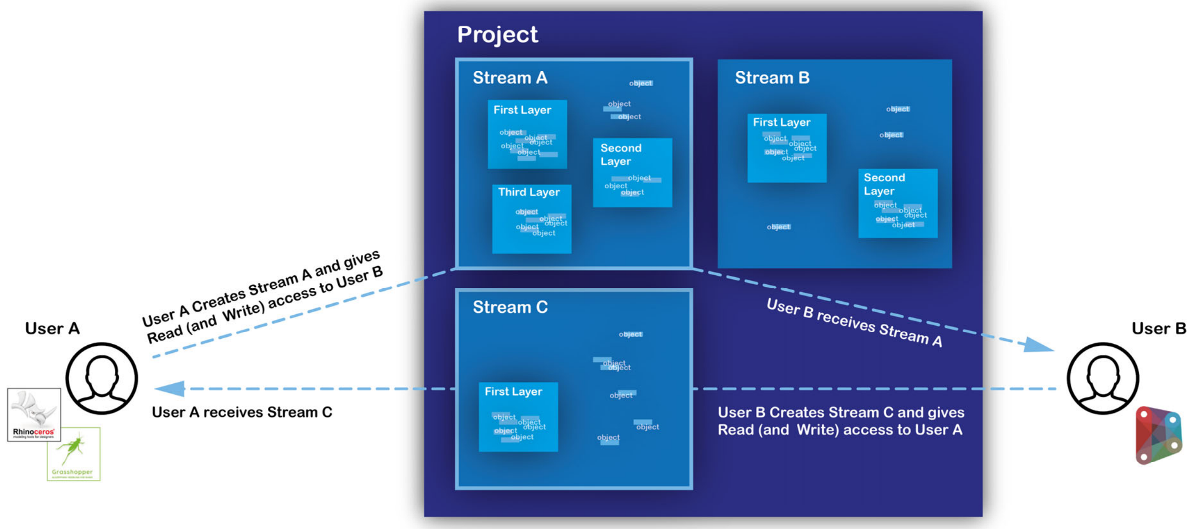

Figure 4. Data exchange protocol between two users.
Streams are created and stored within the Project panel, and contain the project's exchanged data.

provided that they exhibit certain technical characteristics (Speckle, 2020).

Data transfer to and from each end-user is orchestrated by a given Speckle Server, which ensures its availability in the case the original source is offline. Furthermore, the server allows also for efficient updates by leveraging several mechanisms, such as caching, object immutability and partial, differential updates. Speckle implements a discretionary access control model, which gives full control to data authors on how accessible their information is, and with whom. This allows for either fully public or private resources but as well for granular privacy and security settings customised to the roles and needs of each design actor a particular resource is shared with Speckle (2020).

Resources in Speckle are organized in a hierarchical manner as a Work Breakdown Structure (WBS) (see Figure 4) through Objects, Layers (collections of Objects), Streams (collections of Layers and/or Objects) and Projects (collections of Streams). Furthermore, Speckle allows for resources to be enriched with extra metadata such as description, tags, comments, so as to be able to respond to the project's needs and allow for diagonal queries.

Since its inception in 2016, it has been adopted by a large number of progressive AEC companies as a key piece in their digital transformation efforts.

4 SPECKLEVIZ

Via Speckle Streams, users are able to share data from the different existing Speckle clients and plug-ins, which expose a User Interface to both share data (Senders) and receive (Receivers). For example, User A creates a Sender to share Stream A to User B, who creates a Receiver to receive Stream A from User A. As data transfer protocols in Speckle operate in a unidirectional (as opposed to bidirectional) manner, User B would need to create a new Stream (after working upon the data sent by User A via Stream A) to share new data to User A. This simple yet crucial triple protocol (Sender-Stream-Receiver) defines the basis of the Speckle's Activity Network and is illustrated in Figure 5, along with the aforementioned hierarchical directory structure of Speckle (Object, Layers, Streams, Projects). Although one Stream can be contained within multiple Projects, SpeckleViz only renders the activity network happening within a single Project. In other words, SpeckleViz is a tool for illustrating data flows among the project network, a *"network that gets re-initiated for each project"* (Chinowsky et al., 2008, p. 806, Chinowsky et al., 2010, p. 453).

As an architectural project involves a large number of actors working from different software platforms, the above described procedure scales up to form a larger non-linear workflow that is composed of multiple Sender-Stream-Receiver protocols. In general, Speckle Streams are

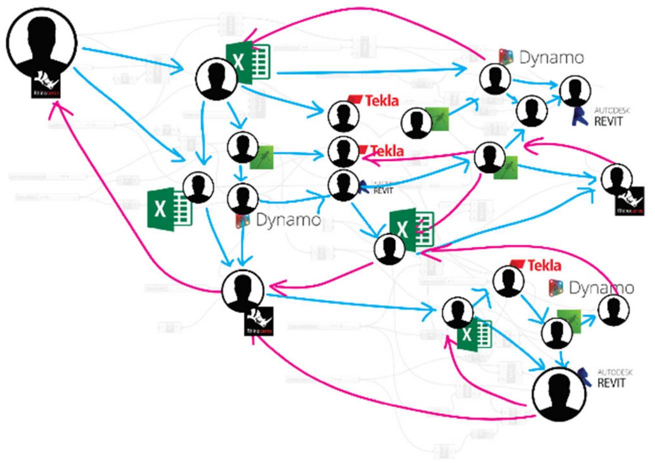

Figure 5. The overall workflow in Speckle is non-linear and contains multiple feedback processes between the different involved trades and software platforms.

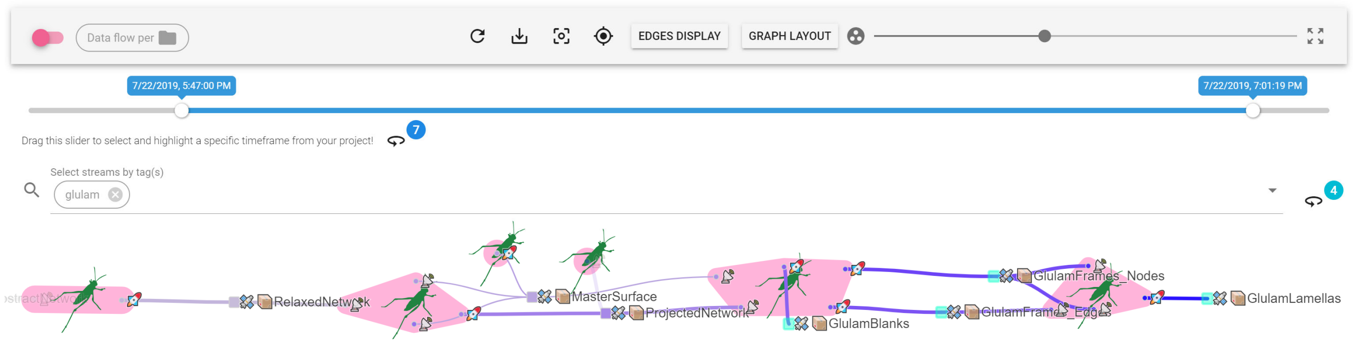

Figure 6. The current SpeckleViz interface (SpeckleViz, 2020).

considered to be single directional flows of data, as this reduces conflicts arising from various sources updating a stream with conflicting information. Although the data flow is deliberatively acyclic (as data transfer mechanisms in Speckle are unidirectional) one could interpret the pattern UserA-StreamA-UserB-StreamB-UserA as a data cycle. In other words, whereas the underlying data transfer in Speckle is acyclic (and can therefore be defined as a Directed Acyclic Graph - DAG), its representation can be, in some instances, perceived as cyclical because it starts and ends with ‘UserA’. Figure 5 illustrates how data transfers would evolve and look like throughout the design process timeline of a building project, amongst multiple disciplines and across different software platforms. The Speckle’s Activity Network described in the next section attempts to visualize these processes by offering an interactive user interface from which the user can query the different created streams.

4.1 Technological Framework

Back-end

On the back-end, the SpeckleViz activity network diagram harvests data through the Application Programming Interface (API) calls using Axios (Axios, 2019). The initial HTTP request takes a Project ID as an input, returning the list of contained streams as a response. New HTTP requests are made to retrieve each stream’s corresponding resources, such as: `_id` (Stream’s ID), `owner` (Stream’s owner), `createdAt` (Stream’s creation time), and `updatedAt` (Stream’s last update), etc. Finally, last HTTP requests are made to get the corresponding Clients (Sender and/or Receiver) resources per Stream, such as: `_id` (Client’s ID), `owner` (Client’s owner), `documentGuid` (Document’s GUID), `documentName`, `createdAt` (Client’s creation time) and `updatedAt` (Client’s last update). Although most of the resources can inform the graph, the main ones used to create its nodes and edges are the Client’s `_id` and Stream’s `_id` properties. The collected resources are formated into a JSON (JavaScript Object Notation) objects which will further feed the graph on the front-end.

Front-end

SpeckleViz (Figure 6) is built upon the Speckle web management admin interface, within the Project tab. As the latter has been designed with Vue.js, an open-source JavaScript framework for building user interfaces and single-page applications (Macrae, 2018), a basic layout has been designed with the same framework in order to host the graph itself, which has been rendered using D3.js (also known as D3, short for Data-Driven Documents) - a JavaScript library for producing dynamic, interactive data visualizations in web browsers (Murray, *2017).*

As Vue.js and D3.js operate on different levels and through different mechanisms, a suitable pattern had to be established to enable the passing of data seamlessly from one framework to the other. In this context, a Vue template has been created to receive the SVG elements from D3.js. For example, the SVG elements `<svg>` `<g>` and `<rect>` elements are added individually rather than through the familiar D3.js method chaining pattern. This allows to dynamically bind these elements to D3.js data within the Vue component, and take advantage of Vue’s reactivity. In general, the graph is generated through d3-force, a D3.js module dedicated to force-directed graph layout using velocity verlet integration. In regards to the styling of the toolbar and control panels, SpeckleViz relies on Vuetify.js (Vuetify, 2020)

4.2 Visualization Features

While circle nodes represent Senders (S) and Receivers (R), square nodes represent Streams. Arrows (or graph edges) represent either data that has been shared to a stream by the user (Receiver to Stream) or data that has been retrieved by a user from a stream (Stream to Sender). The edge’s thickness is proportional to the number of exchanged geometrical objects. Generally, both nodes and edges are coloured according to their respective timestamp: dark blue for the newest created, and light grey for the oldest.

As the graph is force-directed and rendered dynamically, its overall layout might sometimes become too convoluted and not tidy enough to be grasped as a whole. Therefore, several options have been exposed to the end-user in order to manually adapt the graph representation: while the display mode of the graph edges could be switched between three different modes (straight line, arc or diagonal), the force-directed graph layout could be altered in order to force its alignment along the X or Y axis, taking the shape of a tidier

Directed Acyclic Graph (DAG) (see Figure 7). Furthermore, the end-user could at any time during render stop the force-directed simulation.

Senders and Receivers can be grouped either by identical Document GUID or identical Client's owner ID. While the latter is represented by a blue convex hull, the former is visualized by a pink convex hull. The next section will elaborate on an interaction feature enabling the end-user to switch between two modes of representation: user-centred or document-centred.

4.3 Interaction Features

Multiple interaction features have been implemented on the front-end in order to give the end-user a more granular control over the data exposed by SpeckleViz. These features operate on four different levels:

Drop-down menus

Right-clicking on one the graph nodes would display their related drop-down menu. In the case of a Stream, the user can choose between accessing the Stream's information through the Speckle web management admin interface, viewing the Stream within the Speckle viewer interface or accessing the Stream's data available through the API itself. In the case of a Client, the user can retrieve basic information, such as its `_id`, `createdAt` and `updatedAt` properties.

Time frame selection

As specified above, both Streams and Clients expose the createdAt property informing on when the Stream or Client was created. This data has been brought to the front-end of SpeckleViz by integrating a slider within the application, enabling the end-user to select a specific time frame of the project. When dragging the slider, the graph's nodes and links fade out when they are out of range and fade back in when they are in range. Furthermore, the Streams created within the selected time frame are continuously collated and can be visualized altogether inside the Speckle viewer through a dedicated button.

Tab-based queries

In Speckle, Streams can be tagged by the end-user through the web management admin interface. Input tags are then exposed on the API side through the Stream's property `tags`, which is collected on the back-end before rendering the graph in SpeckleViz. On the front-end side, the Vuetify `v-autocomplete` component (Vuetify, 2020) has been integrated, enabling the end-user to select/deselect the existing tags present within the API. The selection dynamically updates the display of the Stream nodes within the graph by highlighting the ones containing at least one tag present within the current selection. Furthermore, the selected tagged Streams are continuously collated and can be visualized altogether inside the Speckle viewer through a dedicated button.

Adaptive representations

While the activity network in Speckle has been so far illustrated from a user-centred perspective (Figure 4 and 5), the related works referenced earlier highlighted workflows that were instead document-centered (Figure 2), or company-centered (Figure 1). Data flows within the AEC sector can therefore be interpreted (and visualized) from different perspectives. In such a context, SpeckleViz attempts to give the end-user the possibility to visualize and adapt its graph from different points of view: the SpeckleViz toolbar exposes a toggle button enabling the user to choose between the *"Data flow per user"* and *"Data flow per document"* modes. Therefore, SpeckleViz provides the users with illustrations of the data flows among the Social Network, as described by Wasserman and Faust (1994), and the inherent Activity Network, as defined by Bauke de Vries (1995). Switching between these two modes dynamically updates the graph that reorganizes its nodes according to the chosen data flow perspective.

Although the main visualization and interaction features have been described above, every single one has been described in more details on the main documentation page of SpeckleViz (SpeckleViz, 2020).

5 CASE STUDY: SPECKLE WORKSHOP AT SIMAUD 2018

As the Project management panel within the web management admin interface is relatively new, not many practices have used it to organize their exchanged Speckle Streams. Instead, Speckle has been used so far in a more informal manner without too much focus on the Project management interface. As a consequence, there exist today only a very few publicly available data sets on which SpeckleViz could be eventually deployed. Therefore, the authors have exploited an existing digital workflow deployed during a workshop focussing on the Speckle platform, conducted at Delft University of Technology on the 4th of June 2018 in the context of the Symposium on Simulation for Architecture and Urban Design (SimAUD) conference. The Streams created and used during this workshop have been reorganized through the Speckle Project panel, serving here as initial data sets to test and deploy the SpeckleViz graph.

5.1 Case study brief and set-up

After an initial introduction to the Speckle communication platform, a parametric modelling workflow of a free-form timber structure developed within the Grasshopper interface has been segregated into eight different computational *"pipelines"* distributed amongst the eight different participants: (1) Global Network Control, (2) Geometrical Optimization (angle maximization between the members), (3) Radius Control and Maximization, (4) Master Surface Control, (5) Blank Mesh Generation, (6) Volume Mesh Generation, (7) Lamella Mesh Generation, (8) Result Overview. Each participant had control over its own local design space. When satisfied with the local design outcome, the workshop participant could communicate the output of

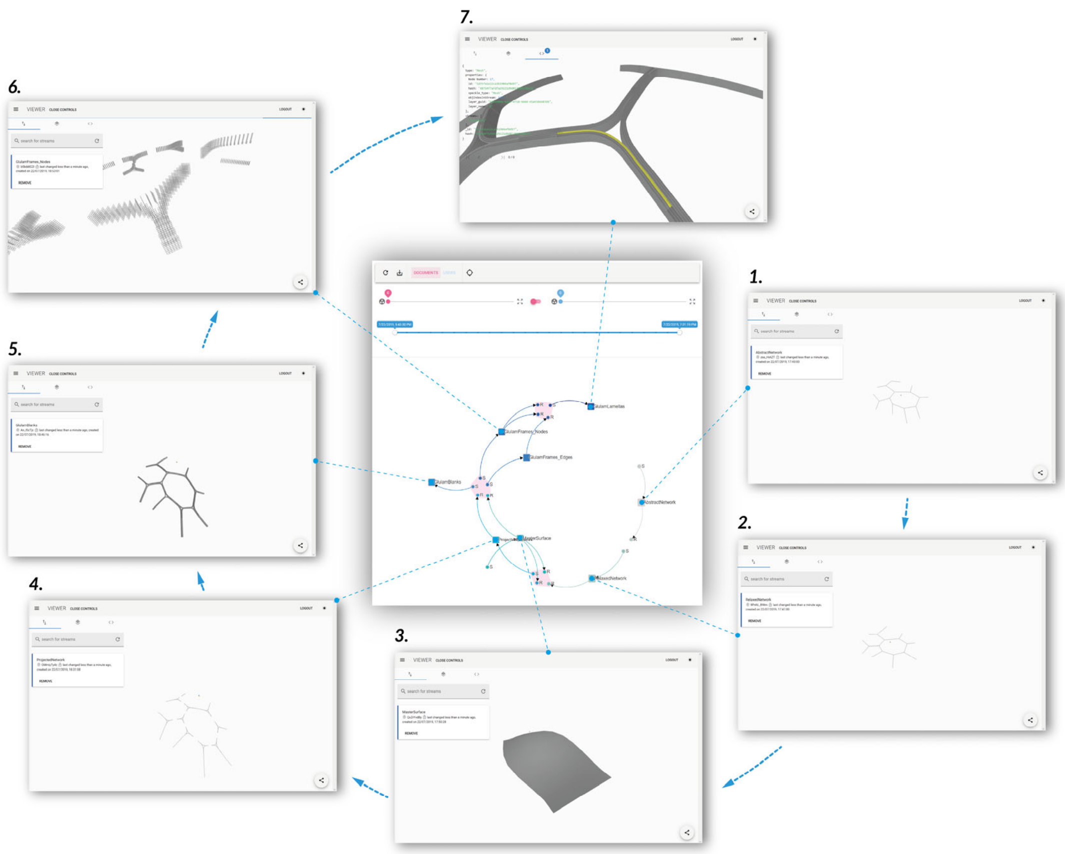

Figure 7. The deployed SpeckleViz interface representing the data flow of the Speckle Workshop at SimAUD 2018

its own computational pipeline to the next design actor, which also had control over its own design space, and so on.

5.2 Case study data collection and analysis

Through the Grasshopper Speckle Client, both Speckle Senders and Receivers were used to seamlessly share design data across all the phases of the design process. The data collected originated directly from the different Rhino-Grasshopper sessions manipulated by the students, and could serve three different purposes:

- **Sharing design ideas:** data could be shared in the sole purpose of exchanging design ideas. This way, students could always log in to the admin interface, explore design possibilities and be inspired by the different models shared by their classmates.
- **Monitoring a Stream's design history:** data could be collected in order to keep track of the design history (enabled by the Speckle platform) of a particular Stream.
- **Keeping track of the project's timeline history:** Finally, data could also be gathered in order to keep track of the chronological evolution of the design process, from the lowest level of detailing to the highest.

The present case study can be seen as an experimental, distributed design chain across all participants, forming the overall design workflow of the workshop. Speckle Senders and Receivers were used to seamlessly share design data across the pipelines. The overall workflow has been represented through the SpeckleViz interface (Figure 7). The end-user (here, a speculative project manager) is able to visualize all the data exchanged during the workshop and access each Stream within the Speckle viewer environment. The interaction features described above were all operable, such as the tag-based query interface. For example, streams tagged as "fabrication" and/or "global design" could be called, collected and visualized within the Speckle viewer.

6 DISCUSSION AND CONCLUSION

This paper introduced the current challenges in collaborative design workflows within the AEC sector and described the SpeckleViz interface, an experimental and work-in-progress web-based interactive visualization tool which aims at representing the activity network (across users or documents) that operates within the open-source Speckle framework.

The SpeckleViz interface has demonstrated how existing open-source frameworks, such has Vue.js (Macrae, 2018), D3.js (Marray, 2017), Vuetify.js (Vuetify, 2020) and Speckle (Speckle, 2020) can be leveraged and orchestrated altogether in order to build a custom application that answers specific needs from the industry (the AEC sector in the present case). The developed application can then feed back to the open source community, and hopefully inspire future open source contributors to develop custom applications built on Speckle.

6.1 Limitations

Maintenance

Currently, Speckle provides five different application integrations (such as Grasshopper3D and Dynamo), and it is expected that this number increases over time as the Speckle community continuously grows and therefore might gain interest in integrating the open-source platform within other specific software packages. Speckle already requires high maintenance as each software package that integrates a Speckle client could modify its .NET API (A or SDK unexpectedly. Consequently, the contributors to the Speckle platform would need to revisit the affected open-source repositories and rewrite specific object model conversions.

Beyond Visualization

The current SpeckleViz interface can mainly be used to monitor a design process, and its interaction features are mostly limited to the inspection of one or multiple streams based on a time range or tag(s). The authors believe that these existing features could be extended beyond simple visualization or Stream inspection by enabling the end-users to directly manage a Speckle project through the graph itself, by adding, deleting, or tagging Streams. More than a visualization tool, SpeckleViz would then act as a data management platform.

6.2 Future work

Although it has not yet been widely adopted by architectural or engineering practices, future collaboration with different AEC companies (e.g. BuroHappold Engineering and Grimshaw Architects) will look precisely at how SpeckleViz could be deployed within and adapted for data exchanges within practice. Future work will also look at how SpeckleViz could represent the data flows beyond a single project environment. Visualizing the data exchange across multiple projects, companies and/or servers remain open questions that still need to be addressed. Enabling adaptive display strategies to focus at different levels of representation (collapsing the nodes belonging to the same document or same user into one) is another aspect that needs to be tackled. Finally, further research will investigate more in depth the data available within the Speckle API on the back-end to increase analytics and give richer insights to the end-user on the front-end.

REFERENCES

1. Autodesk Revit is a Building Information Modelling (BIM) software for architects, engineers, designers and contractors.
2. Axios, A promise based HTTP client for the browser and node.js. https://github.com/axios/axios. 2020.
3. Booch, G., Rumbaugh, J. & Jacobson, I. *The Unified Modeling Language User Guide.* (2nd ed.) Addison-Wesley Professional, 2005.
4. Chinowsky, P., Diekmann, J. & Galotti, V., Social network model of construction. *Journal of construction engineering and management*, 134 (2008), 804-812.
5. Chinowsky, P.S., Diekmann, J. & O'brien, J., Project organizations as social networks. *Journal of Construction Engineering and Management* (2010), 452-458.
6. Davies, A. *Project management for large, complex projects*. London, United Kingdom: The Bartlett School of Construction and Project Management (2019).
7. De Vries, B. Message Development in the Building Process. *Modeling of Buildings through their Life-Cycle. Proceedings of the CIB w78 Conference.* Standford (1995) 467-479.
8. Dynamo is a graphical programming interface within Revit.
9. Flux, 2018. Retrieved from http://flux.io/ (accessed before the 31st of March 2018).
10. Grasshopper3D (typically abbreviated Grasshopper) is a visual programming language and environment that runs within Rhino3D.
11. Macrae, C. *Vue.js – Up and Running*. O'Reilly, 2018.
12. Murray, S. *Interactive Data Visualization for the Web - an Introduction to Designing with D3*. (2nd ed.) Sebastopol: O'Reilly, 2017.
13. Ringley, B. San Francisco Computational Design User Group, June 2017: *What is the point of using Dynamo?* (https://www.youtube.com/watch?v=y6N1ICoFoyU)
14. Rhinoceros (typically abbreviated Rhino, or Rhino3D) is a commercial 3D computer graphics and computer-aided design (CAD) application software developed by Robert McNeel & Associates. Available from https://www.rhino3d.com/ (accessed on the 17th of March 2020).
15. Speckle, 2020. Retrieved from http://speckle.systems (accessed on the 17th of March 2020).
16. SpeckleViz, 2020. Retrieved from https://speckle.systems/docs/web/speckleviz/ (accessed on the 17th of March 2020).
17. Van Der Heijden, R., E. Levelle, and M. Reise. Parametric Building Information Generation for Design and Construction. In *Proceedings of the 35th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA 2015)*. 417–430.
18. Vuetify, A Material Component Framework for Vue.js. https://github.com/vuetifyjs/vuetify. 2020.
19. Wasserman, S. and Faust, K. *Social network analysis: Methods and applications.* Cambridge, United Kingdom: Cambridge University Press, 1995.
20. White, S. A. *Introduction to BPMN.* BPTrends, July 2004.
21. Ziegler, B. P., Praehofer, H. & Kim, T. G. *Theory of Modeling and Simulation*. (2nd ed.) Academic Press, 2000.