WEB-BASED DISTRIBUTED DESIGN TO FABRICATION WORKFLOWS

PAUL POINET[1], DIMITRIE STEFANESCU[2] and
ELENI PAPADONIKOLAKI[3]
[1,2,3]*University College London - Bartlett School of Construction and Project Management*
[1,2,3]*{p.poinet|d.stefanescu|e.papadonikolaki}@ucl.ac.uk*

Abstract. As architectural design projects tend to tackle larger scales and become more complex, multiple involved actors often need to work from different remote locations. This increased complexity impacts the digital design-to-fabrication workflows that become more challenging, as each actor involved in a project operates on different software environments and needs to access precise fabrication data of specific design components. Consequently, managing and keeping track of design changes throughout the design-to-fabrication workflow still remains a challenge for all actors involved. This paper discusses how this challenge can be tackled through both Speckle, a complete open source data platform for the Architecture, Engineering and Construction (AEC), and SpeckleViz, a custom web-based interactive Activity Network Diagram (AND) built upon Speckle. SpeckleViz continuously maps data transfers across design and building processes, enabling the end-users to explore, interact and get a better understanding of the constantly evolving digital design workflows. This is demonstrated in this paper through a computational design and digital fabrication workshop conducted at the Centro de Estudios Superiores de Diseño de Monterrey (CEDIM), during which an integrative, file-less collaborative design workflow has been set through Speckle, connecting different Rhino-Grasshopper sessions acting as discrete computational design pipelines.

Keywords. Collaborative Workflows; Distributed Design; Activity Network Diagram; Data Flow.

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: *"Many infrastructure projects in the UK now recognize the need for solutions that are designed to deal with the specific challenges involved in planning and executing large, complex projects. (...) Over the past decade,* *many of the UK's largest and most complex infrastructure projects have abandoned traditional delivery models"* (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 widely available.

Apart from being able to model complex geometry, 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 the AEC sector, current design processes are still segregated, and laborious manual interventions sometimes become a daily routine (Burry and Holzer 2009). To tackle this issue, custom management and visualization tools enabling better understanding and curation of complex projects Activity Network Diagrams (AND) 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. To this end, this study aims to address this need by presenting a web-based open-source interoperability framework Speckle (2020) that increasingly gains traction in the AEC. Built upon Speckle, SpeckleViz (2020) an interactive AND, helps end users to get an overview of data flows in a specific Speckle Project. This paper is divided into five sections after this introduction: description of Speckle framework, description of SpeckleViz interface, empirical testing in a case study of "Piped Assemblies" workshop, where both Speckle and SpeckleViz were deployed, discussion section and section with concluding remarks.

2. Speckle open-source data framework

Speckle (2020) is a unique, complete open-source data framework for the AEC that differs from commercial web-based interoperability platforms. It was originally developed at University College London (UCL) in 2016 by Dimitrie Stefanescu (Stefanescu 2020). Unlike the existing AEC standard Industry Foundation Classes (IFC) that characterizes most of Building Information Modelling (BIM) software available today, Speckle does not impose a predefined topology of communication patterns (or specific object model), but rather allows for the emergence of meaningful data-driven dialogue amongst different actors involved in the design process. Contrary to the current single monolithic BIM model paradigm that argues in favor of centralizing all data coming from different trades involved in the design process of an architectural project, Speckle embraces and acknowledges the existing decentralized design processes by providing an open source infrastructure to better manage and curate them.

Practically, Speckle currently offers a set of Computer Aided Design (CAD) application integrated clients (Grasshopper, Dynamo, Rhino, Blender, Revit and GSA), which support seamless data exchanges through "Streams". In Speckle, a Stream is essentially a collection of geometrical objects that has been sent to or received from a Speckle client. As the data exchanges happen informally across users working remotely from the different mentioned software platforms, SpeckleViz - a web application for Speckle - has been implemented, in order to obtain a clear visual representation of the overall data flow for a specific project.

3. SpeckleViz data flows visualization interface

SpeckleViz is an AND developed in Speckle to get a better understanding of data flows in a Speckle Project across multiple Speckle Users (users logged in the Speckle Admin interface), Speckle Streams (collections of Speckle Objects shared across multiple users) and documents (the software environments from which data has been sent or received via Speckle). The interface is situated at the bottom of a Speckle Project page and directly reflects how Speckle Streams are organized and relate to each other within a same Speckle Project. The SpeckeViz interface (see Figure 1) is divided into two main parts: the control interface, composed of the main toolbar, the time range slider and the tag query selector, and the graph canvas. The next paragraphs summarize the main visualization and interaction features of SpeckleViz and the complete documentation of the interface is available on its dedicated documentation page available within the main Speckle website (SpeckleViz 2020).

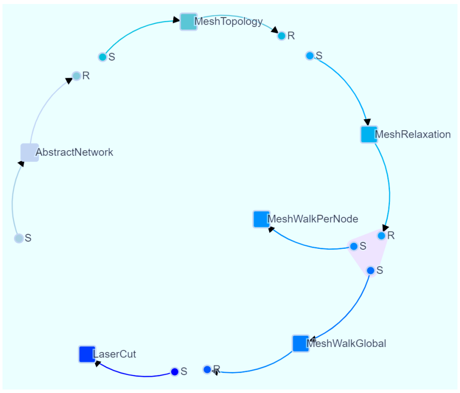

Figure 1. The SpeckleViz interface can be found at the bottom of a Speckle Project page.

3.1. SPECKLEVIZ VISUALIZATION FEATURES

Similarly to any graph (Biggs et al., 1976), SpeckleViz consists of nodes and edges. First, circle nodes represent Speckle Senders or Speckle Receivers, square nodes represent Speckle Streams. Second, 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 edges' thickness is proportional to the number of exchanged geometrical objects. Generally, both nodes and edges are colored according to their respective *createdAt* timestamp: dark blue for the newest created, and light grey for the oldest. Senders and Receivers can be grouped either by identical Document GUID (Globally Unique Identifier) 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.

3.2. SPECKLEVIZ 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:

- **Dragging nodes:** Nodes can be dragged through the canvas by holding left click on a specific node.
- **Zooming/Panning:** It is possible to zoom in or zoom out by using the scroll wheel, and pan by holding left click on the background canvas and moving the mouse around it. If the graph disappears, clicking on the GPS button situated on the toolbar will re-center the graph.
- **Switching between data flow types:** The clients are grouped either by Document GUID or User GUID. The SpeckleViz toolbar exposes a toggle button enabling users to choose between the *"Data flow per user"* and *"Data flow per document"* modes. Thus, SpeckleViz provides users with illustrations of data flows among the Social Network, as described by Wasserman and Faust (1994), and the inherent AND, as defined by Bauke de Vries (1995). Switching between these two modes dynamically updates the graph that reorganizes its nodes according to chosen data flow perspective.
- **Expanding/Collapsing clusters:** The slider located on the right side of the toolbar enables the user to expand or collapse the clusters described above. This might be useful to disentangle the wires and zoom in on a particular cluster.
- **Inspecting a specific stream:** Right clicking on a Stream will trigger a context menu from which it is possible to: (1) inspect the Stream within the Speckle Viewer, (2) inspect the Stream within the Speckle Admin, (3) inspect the Stream's raw data and (4) inspect the Stream's related client's raw data.
- **Inspecting a time frame:** 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.
- **Refreshing the graph:** If the project is modified (e.g. through the addition or removal of a Speckle Stream), the user will need to hit the refresh button situated on the left side of the toolbar, as the graph won't listen for changes and therefore won't update automatically.
- **Tag-based queries:** In Speckle, Streams can be tagged by the end-user through the web management admin interface. In SpeckleViz, streams can be selected by tag(s) - which were previously set through the management interface - within the search bar situated below the time range slider. 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.

4. Case study: "Piped Assemblies" Workshop at CEDIM

This study uses a case study to provide empirical data on the operation of Speckle and SpeckleViz and illustrate their function. SpeckleViz has been tested against data generated from design to fabrication during the "Piped Assemblies" Workshop conducted at CEDIM during 26 November to 7 December 2018. This workshop addressed the challenge of introducing state of the art web-based collaborative computational design workflows to undergraduate students in architecture, through the design and fabrication of a free-form networked structure made of laser-cut polypropylene plastic strips.

4.1. 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:

- **Exchanging 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 and explore and be inspired by the different models shared by their classmates.
- **Keeping track of the project's timeline history:** 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 (as defined in sections 4.2.1 and 4.2.2). In this specific case, SpeckleViz helped the users to visualize and inspect the evolutionary process of the data flow which is passed from one Stream to another, through time.

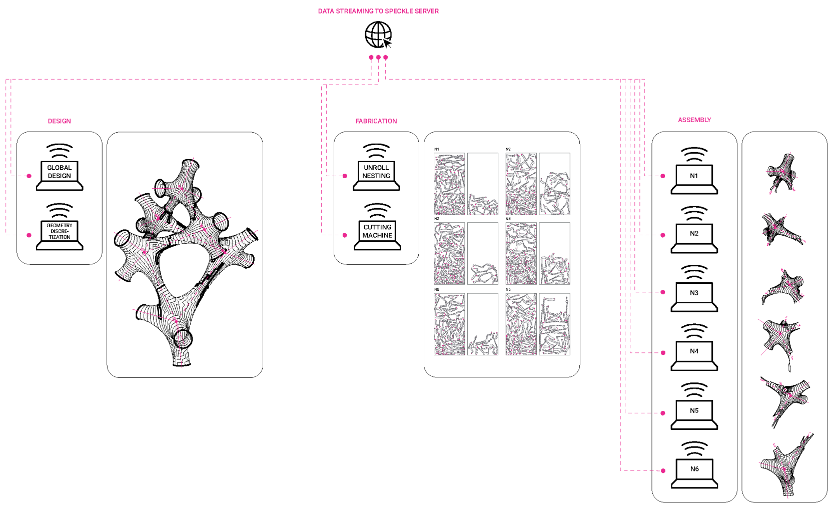

Figure 2. Overall design to fabrication workflow of the Piped Assemblies workshop.

4.2. DESIGN-TO-FABRICATION WORKFLOW

The design-to-fabrication workflow of the Piped Assemblies workshop had to be both flexible enough to let the students explore different design options, and to embed rationality (Attar et al. 2010) as well as multiple constraints for further fabrication and assembly feasibilities. While the students were subdivided into three different groups responsible respectively for design, fabrication and production, the overall design to fabriaction workflow (see Figure 2) has been segregated into 7 distinct computational pipelines - or remote concurrent shared parametric models (Burry and Holzer 2009) - which are all connected through Speckle Streams:

4.2.1. Design Generation and Embedded Rationality

- **(1) Definition of the overall abstract network.**
- **(2) Base mesh generation.** This step was particularly crucial to generate a design which had to be consistent enough to be fabricated and assembled with differentiated polypropylene strips. The mesh resolution, the curvature relaxation and the planarity of each mesh face all had to be considered to evaluate the feasibility of the design.
- **(3) Mesh relaxation and planarization.** Once a consistent global mesh has been generated in the previous step, a dynamic relaxation has been performed using Kangaroo 2 (Piker 2020), a live physics engine for interactive simulation, form-finding, optimization and constraint solving. The relaxation used here constrained all the mesh quads to remain planar, ensuring further developability for laser cut fabrication.

4.2.2. Data Generation for Fabrication

- **(4) Strips generation.** Here, a mesh-walk algorithm has been developed. The python library NetworkX (2020) allowed to convert the relaxed mesh into a graph, and run the Dijkstra's Shortest Path First algorithm which could find the shortest path between two mesh face center points. Although not perfectly geodesic, such path tends to minimize the curvature of each strip. This particular design process was iteratively computed through a feed-back loop: after the generation of one strip, the latter was "removed" from the global mesh, and the Dijkstra's Shortest Path First was re-computed onto the "remaining" global mesh. The feedback loop stopped once the global mesh was completely transformed into individual strips. From this step, the overall discretized structure has been segregated into six different nodes.
- **(5) Labelling, unrolling and sorting of the strips.** From this step, the students had to orchestrate themselves into six different teams to Separate the Concerns (Dijkstra 1970) and fabricate separately the six different nodes of the structure, thus reducing complexity of the future assembly process. For each node, the processes described from (5) to (7) intertwined themselves in order to optimize the timeline of the fabrication process. Indeed, while a team focuses on the laser cutting process, another team does not have to wait for all the pieces of the entire structure to be ready and can start assembly as soon as a set of strips

necessary to form a node is ready. The same goes for the assembly process between multiple nodes, which can start as soon as two neighboring nodes are ready to be connected.

- **(6) Nesting of the strips** through OpenNest (Vestartas 2020) - a plug-in for Grasshopper - and generation of their respective connection tabs around the edges. During the nesting process, the strips were distributed amongst 12 boards of 122*244 centimeters. Finally, the tabs for the rivet connections were generated by means of custom offsets.
- **(7) Exporting the fabrication data** via .dxf files for machine-ready deliverables and wireless transfer to the laser cutter at STM Robotics (laser cutter fabricator and sponsor of the workshop).

4.2.3. Fabrication and Assembly

After being laser cut at STM Robotics, all the polypropylene plastic strips had to be manually sorted both by their respective length and index number (engraved into which element) to ease the further fabrication process. This design-to-fabrication workflow described above enabled the participants to track each strip and detect their neighboring pieces, within and between each node. The six nodes have been constructed parallelly but assembled one after the other (one neighbor at a time), facilitating the reach of the connecting tabs from both inside and outside the nodes during the manual placement of the rivets. The final assembled physical prototype has been hanged with nylon threads in the main entrance of CEDIM, between the first and the ground floor through the mezzanine aperture (see Figure 3). The tension of the structure was adjusted manually and empirically by calibrating the tensile force and anchor location of each thread.

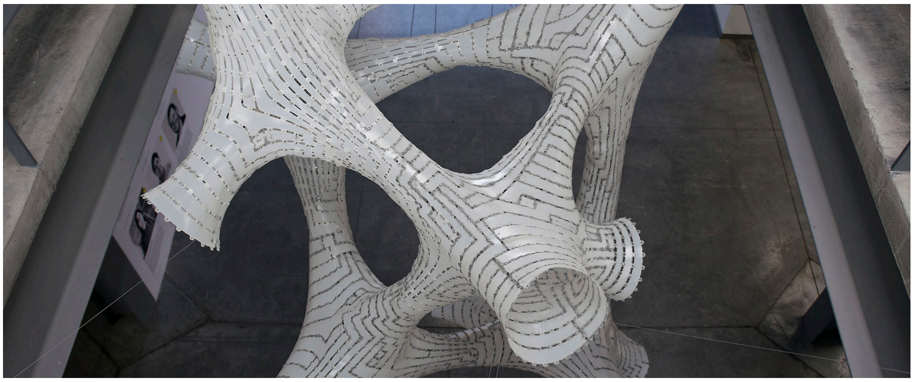

Figure 3. Final installation suspended above the main entrance of CEDIM.

4.3. WEB-BASED COLLABORATION THROUGH SPECKLE STREAMS AND SPECKLEVIZ

After describing the design-to-fabrication workflow deployed in the workshop, reflection on the communicative and collaborative strategies deployed throughout the project follows. Using a Speckle server to gather geometrical data allowed flexibility in the design and fabrication process as team members were not required to colocate, but could instead spread across the various locations to support design, fabrication and assembly of the structure, and work remotely from those: (1) the laser cutter at STM Robotics for fabrication, (2) the university's workshop for assembly and (3) the exhibition space for installation. Through the Grasshopper Speckle clients, both Senders and Receivers were used to seamlessly share design data across all design actors, software platforms and across pipelines.

During each design transaction, and via Speckle and SpeckleViz, the students were able to inspect and enrich geometric information with metadata needed for their specific subtasks, thus maintaining a high degree of information efficiency and little overhead in terms of data payload, as the minimum required information could be seamlessly shared via Speckle Streams, without the need of converting and exporting entire files. Multiple informational Streams were created, aggregated and throughout the process (see Figure 4). In the present case study, each Stream geometrically depends on each other, and the overall workflow goes from the lowest level of detailing to the highest:

- 1. Definition of the overall abstract network (generated by the design group).
- 2. Base topological mesh design generation (generated by the design group).
- 3. Mesh relaxation (generated by the design group).
- 4. Strip segments generation (inspected by the fabrication groups).
- 5. Unrolling and nesting (generated and inspected by the fabrication groups).

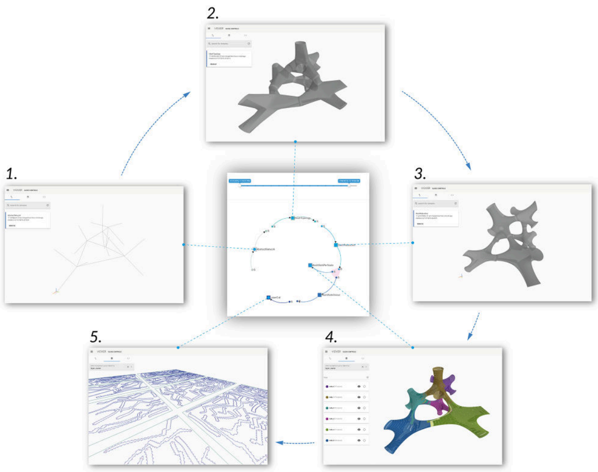

Figure 4. Through the SpeckleViz interface, the user can visualize the data exchanges and access each Stream within the Speckle viewer, from the design to the fabrication stages.

Here, the design Streams in Speckle maintained their relevance and usefulness both in the later fabrication stages, during laser cutting and labelling processes, as well as during assembly, by being able to visually relate parts to the whole, interactively, and on-site.

5. Discussion

The present case study demonstrated that Speckle could orchestrate a complete workflow, in which data are sent seamlessly from and to different Streams, and thus throughout the whole design process of an architectural project in an academic context. Furthermore, SpeckleViz enhances design collaboration at the project by allowing Speckle users to visualize and inspect the data flow taking place at any stage in the design process through AND. The workshop case described above, demonstrated that users could easily use SpeckleViz to trace back the project timeline history and retrieve specific Streams that were previously created and shared.

Both Speckle and SpeckleViz enable a seamless flow of data across different design processes that were previously segregated (Burry and Holzer 2009) and offers new perspectives for the design to fabrication workflows of large-scale and complex architectural design projects. This is particular important for contemporary projects that lack custom solutions (Davies 2019). Whereas the present case study happened in the context of an academic workshop for undergraduate students in architecture, the authors believe that these particular design methodologies and new ways of sharing data through the Speckle platform could also be deployed in practice, as they present great potential for seamless collaboration strategies. Such design methodologies offer new ways of sharing data through the Speckle platform and present great potential for seamless collaboration strategies. For example, the Building Information Generation (Van der Heijden et al. 2015) workflow, a modelling strategy deployed by Front Inc., that consists of a strategic alternation between the generation of objects in Grasshopper and their subsequent storage and classification within staged Rhino3D models, could largely benefit of the Speckle platform which would enable a live connection between the different deployed Rhino3D and Grasshopper clients, as well as SpeckleViz which would enable the Speckle users to visualize and inspect the data flow throughout the whole design process.

6. Conclusion and Outlook

This paper puts forward a new, prototypical way through which information modelling can be employed in an agile manner in fabrication processes. Through the design process and physical outcome of the digital fabrication workshop conducted at CEDIM, we show that a distributed, informal communication process can evolve into a highly efficient and lean workflow that maintains the necessary precision and coordination for precision manufacturing and assembly. Future work will look more particularly at how SpeckleViz could represent data flows beyond a single project environment. Visualizing data exchanges across multiple projects, companies and/or servers would be next steps towards streamlining design and fabrication in the AEC.

Acknowledgements

The authors would like to thank Djordje Stanojevic, Yessica Mendez, David Durán, the workshop sponsors and the students involved in the Piped Assemblies workshop at CEDIM. The SpeckleViz interface has been developed during the "AEC Delta Mobility" project which has received funding from InnovateUK under the competition "Increase Productivity, Performance and Quality in UK Construction" (proj. no. 104799). Speckle was originally developed at The Bartlett School of Architecture as part of the InnoChain project, which received funding from the European Union's Horizon 2020 research and innovation programme under the Marie-Sklodowska-Curie grant agreement No 642877.

References

"Piped Assemblies" : 2019. Available from <http://blog.cedim.edu.mx/arquitectura/piped-assemblies/> (accessed 28th of January 2020).

"Rhinoceros (typically abbreviated Rhino, or Rhino3D) is a commercial 3D computer graphics and computer-aided design (CAD) application software developed by Robert McNeel & Associates" : 2020. Available from <https://www.rhino3d.com/> (accessed 28th of January 2020).

"Grasshopper is a visual programming language and environment that runs within the Rhino3D." : 2020. Available from <https://www.grasshopper3d.com/> (accessed 28th of January 2020).

"SpeckleViz documentation page" : 2020. Available from <https://speckle.systems/docs/web/speckleviz/> (accessed 28th of January 2020).

"NetworkX homepage" : 2020. Available from <https://networkx.github.io/> (accessed 28th of January 2020).

Balaji, S.B. and Murugaiyan, S.M.: 2012, Waterfall vs V-Model vs Agile: A Comparative Study on SDLC, *International Journal of Information Technology & Business Management*, **2**(1), 26-30.

Biggs, N., Lloyd, E.K. and Wilson, R.J.: 1976, *Graph Theory 1736-1936*, Oxford University Press, New York, USA.

Burry, J.B. and Holzer, D.H.: 2009, Sharing Design Space: Remote Concurrent Shared Parametric Modeling, *Proceedings of eCAADe 2009*, Istanbul Technical University, Faculty of Architecture, 333-340.

Davies, A.D.: 2019, *Project management for large, complex projects*, Association for Project Management, Princes Risborough.

Dijkstra, E.W.D.: 1970, Notes on Structured Programming, *EUT report. WSK, Dept. of Mathematics and Computing Science*, **70**(3), 1-84.

Van Der Heijden, R., Levelle, E.L. and Reise, M.R.: 2015, Parametric Building Information Generation for Design and Construction, *Proceedings of ACADIA 2015*, Cincinnati, Ohio, United States, 417-430.

Mirtschin, J.M.: 2019, "Revit IFC User Meeting" . Available from <https://www.youtube.com/watch?v=1fJbQ2PmhqA> (accessed 28th of January 2020).

Neijur, A.N. and Seinfeld, K.S.: 2016, Ivy: Bringing a Weighted-Mesh Representation to Bear on Generative Architectural Design Applications, *Posthuman Frontiers: Data, Designers, and Cognitive Machines, Proceedings of the 36th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*, Ann Arbor, 140-151.

Piker, D.P.: 2020, "Kangaroo2 download page" . Available from <https://www.food4rhino.com/app/kangaroo-physics/> (accessed 28th of January 2020).

Stefanescu, D.S.: 2020, *Alternate Means of Digital Design Communication*, Ph.D. Thesis, UCL.

Vestartas, P.V.: 2020, "OpenNest homepage" . Available from <https://www.food4rhino.com/app/opennest/> (accessed 28th of January 2020).

De Vries, B.D.V.: 1995, Message Development in the Building Process, *"Modeling of Buildings through their Life-Cycle", Proceedings of the CIB w78 Conference*, Standford, 467-479.

Wasserman, S.W. and Faust, K.F.: 1994, *Social network analysis: Methods and applications*, Cambridge University Press, Cambridge, United Kingdom.