AUTODESK UNIVERSITY 2015

AS11178

Dynamo for Integrated Computational Engineering

Erik Narhi
BuroHappold Engineering

Learning Objectives

- Gain knowledge of the usefulness of an integrated modeling environment
- Learn how to draw on strengths from multiple modeling platforms
- See practice-based examples of dissemination, documentation, and reusability of computational engineering tools
- Discover examples of customized Dynamo workflows in action

Description

Integrated cross-disciplinary engineering requires integrated modeling. In turn, this necessitates a computational environment that promotes automation and facilitates collaboration and interoperability. At BuroHappold Engineering, Revit software is the principal Building Information Modeling (BIM) software, used across the organization for model coordination and deliverable production. Based on this, BuroHappold has developed global protocols for interdisciplinary collaboration, information exchange, and retention and reusability of computational knowledge. Drawing on the Revit software database, a visual programming environment such as the Dynamo extension platform can act as conduit and integrator between modeling, analysis, and production. We can use Dynamo extension's built-in package manager in conjunction with resources such as a company intranet for dissemination of workflows as well as for version and dependency tracking. The effectiveness of such sharing of computational protocols and standards is a major component for collaboration in design practice.

Your AU Expert

Erik Narhi is a Technical Designer at BuroHappold Engineering working at the intersection of engineering, architecture and computational design. Erik received his M. Arch from USC in 2014. While at USC, Erik conducted research on optimization of regolith lunar structures as part of a NASA NIAC grant and served as a Grasshopper and Scripting Advisor for multiple Graduate Studios. At BuroHappold, Erik uses his experience with computational methods, design, and analysis to contribute to the efficient delivery of elegant solutions to complex engineering challenges.

The Value of an Integrated Modeling Environment

Introduction

A wide range of software is used in the AEC industry. Each piece of software contains strengths, and as a model travels through various programs (Rhino, Revit, Robot, etc.) a large array of data is generated, lost, regenerated, rationalized, and regenerated all over again. This leads to large amounts of rework and a loss of some of the benefit that each program provides in the first place. A sophisticated integrated modeling environment allows designers to tap into the strengths of all software used without suffering the drawbacks and data loss that can occur when exporting, importing and rebuilding repeatedly.

What is an Integrated Modeling Environment?

An integrated modeling environment is an environment in which the user can access the tools and methods of many different programs in order to tap into each of their strengths individually. It is a Swiss Knife, where each individual tool is available for use when its application is most appropriate to the design process. Developing a robust IME depends on interoperability, sophisticated data management, and a means of understanding the model's components in a software agnostic manner so that each program can individually access and contribute to the overall datascape generated by the design process. With BIM already treating a building model as a database, a strong IME provides the ability to expand upon the power of BIM, ultimately providing a better understanding of a building's behavior earlier in the design process, thereby delivering better design solutions more efficiently.

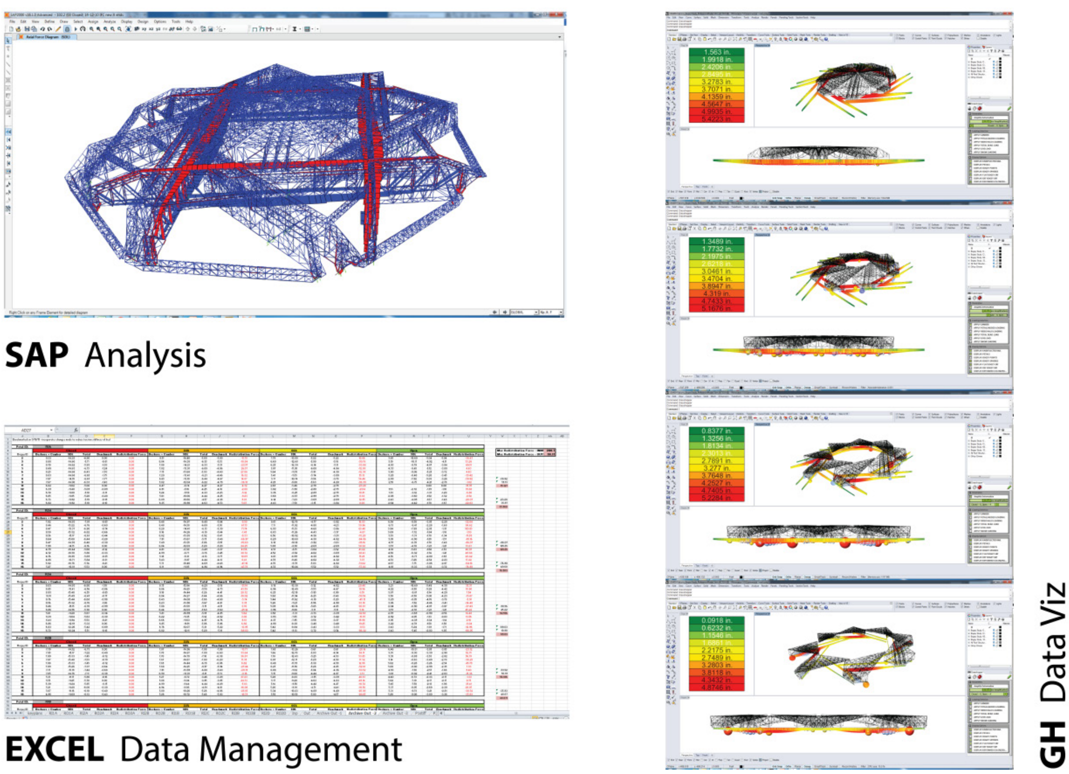

FIGURE 1: TAPPING INTO MULTIPLE PROGRAM STRENGTHS – SAP / EXCEL / GRASSHOPPER

The Conduit for Integration

A key element of developing an integrated modeling environment that requires careful consideration is the platform that will support the environment. If the platform presents issues (licensing, complexity, interoperability, updates breaking functionality) it can cause major issues down the line. Dynamo presents a strong integration platform due its nature; it is open source, databased, and easy to pick up due to its visual programming interface. In addition, package development for Dynamo provides the potential for a firm-wide C# repository containing custom code. This provides the means for a single code share, allowing for simple simultaneous development of tools on the back end and utilization of tools by a wider user base on the front end.

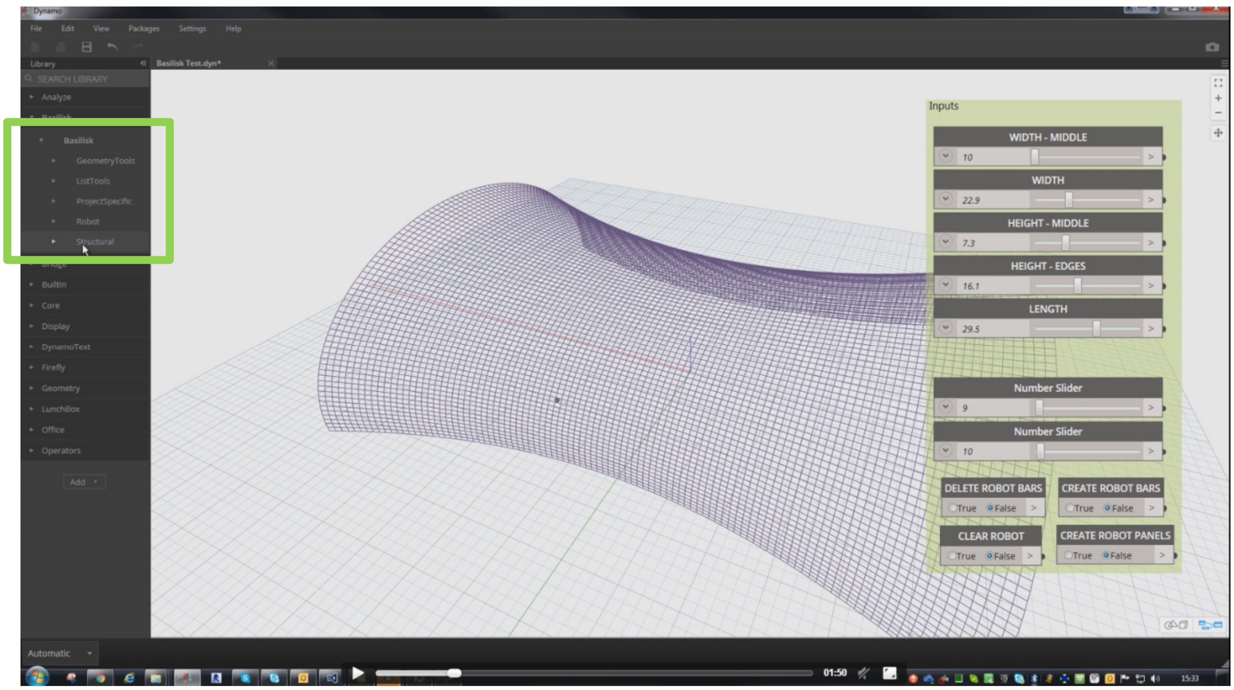

FIGURE 2: THE DYNAMO INTERFACE WITH CUSTOM PACKAGE FROM C# HIGHLIGHTED

Prior to Dynamo, BuroHappold employed a variety of tools developed in-house on a project basis. These tools accommodated the need for an Integrated Modeling Environment on projects too complex for traditional non-integrated workflows, but lacked the modularity and reuse afforded by more accessible integration tools. With Dynamo, it is possible to achieve improved reusability of tools and workflows while also making the environment more accessible to individuals who may not have familiarity with coding. Ultimately, Dynamo can shift the integrated modeling environment from a black-box based series of interoperability tools and plugins towards a light, accessible interface that users of all levels can engage with.

Different Programs == Different Strengths

Interoperability

In order to make the most of the different strengths that each software possesses, interoperability is key. Thanks to the multitude of plugins, tools, and OOTB support for interoperability that are available, a large amount of interoperability can be achieved before taking on any in house development or coding. However, the full potential of interoperability is best realized by customizing your tools in a way that applies each program's individual strengths and adapts them to your typical workflow and deliverables. These custom interoperability tools have some basic requirements to ensure maximal value:

- Tools must be robust. The front end time cost of developing in house tools is only worthwhile if the tool remains useful after its initial application.
- Tools must maintain data between programs in order to minimize rework and maximize the value provided by each step in the design process
- Tools must be well-developed with standardized naming and layout conventions. The simpler the tool the better. This avoids the "Black-Box" phenomenon and ensures that tools may be customized or extended later.
- Ideally, tools should all revolve around one common language. It is much easier to develop a common language and a set of translator tools than to develop direct links between all programs (See Figure 3).

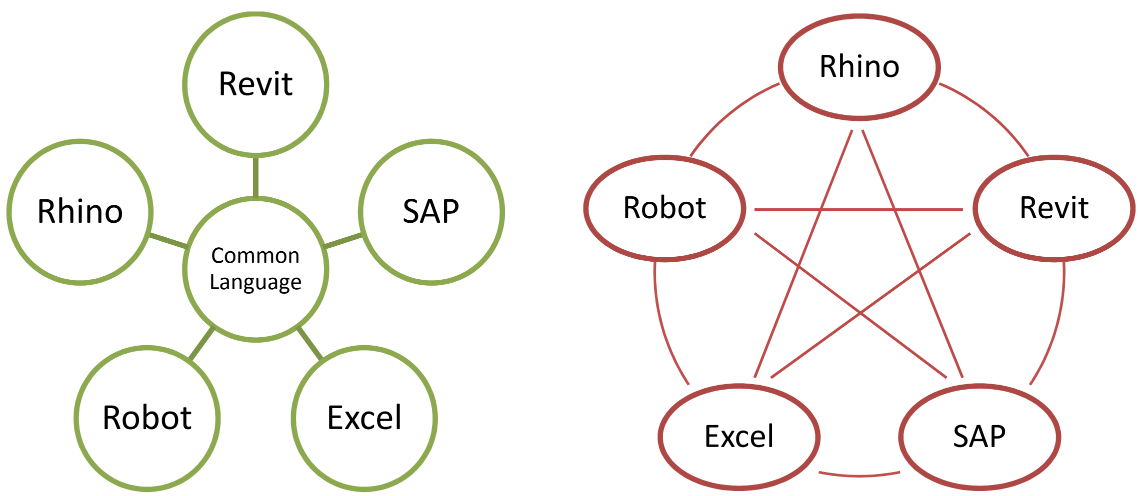

FIGURE 3: SPOKE MODEL VS. WEB MODEL – SPOKE REQUIRES FEWER TOOLS AND MAINTAINS A COMMON SOFTWARE-AGNOSTIC LANGUAGE

The Datascape

Each project generates a large datascape that is rarely fully tapped into. Interoperability maximizes data retention as well as data feedback. A larger datascape to access and a means of understanding this data with tools developed for use in the Integrated Modeling Environment provide stronger understanding of the implications of design decisions, leading to superior design solutions. Given BuroHappold's extent of analysis performed on each project (as an Integrated Design Engineering firm, we often generate millions of data points for a single stage of a single project), a heightened understanding of the data we generate can provide massive return in terms of building performance, occupant satisfaction, design quality and cost.

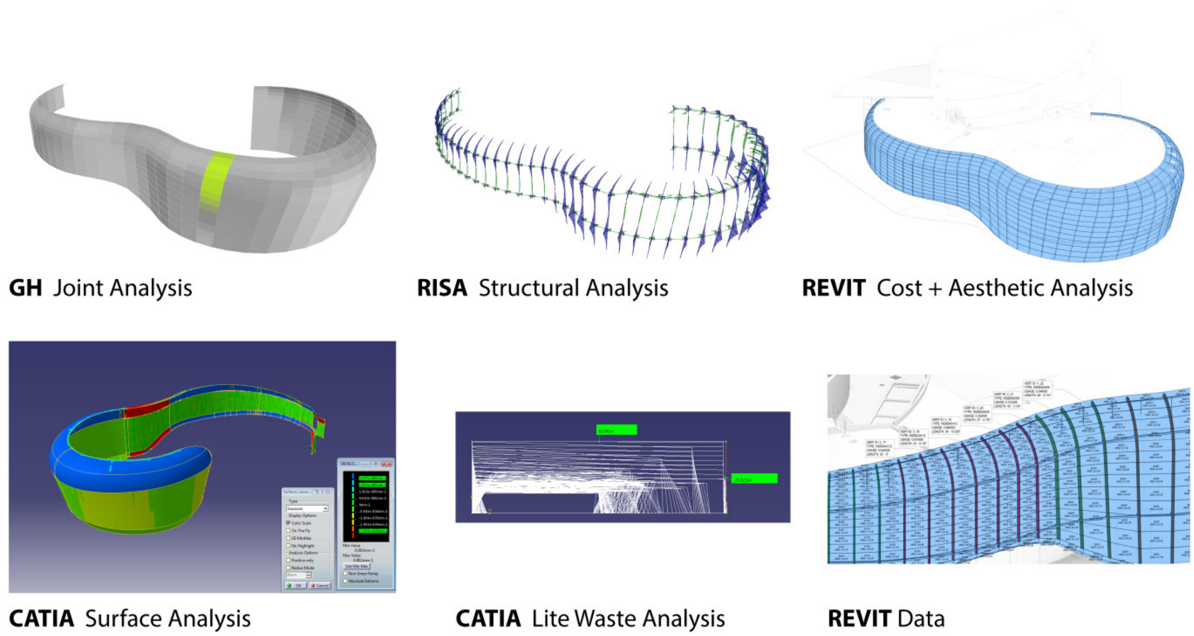

FIGURE 4: AQUARIUM OF THE PACIFIC – INTEROPERABILITY ALLOWED FOR IMMEDIATE UNDERSTANDING OF HOW INDIVIDUAL DESIGN DECISIONS EFFECTED MULTIPLE DESIGN DRIVERS

We Have the Tools…Now What?

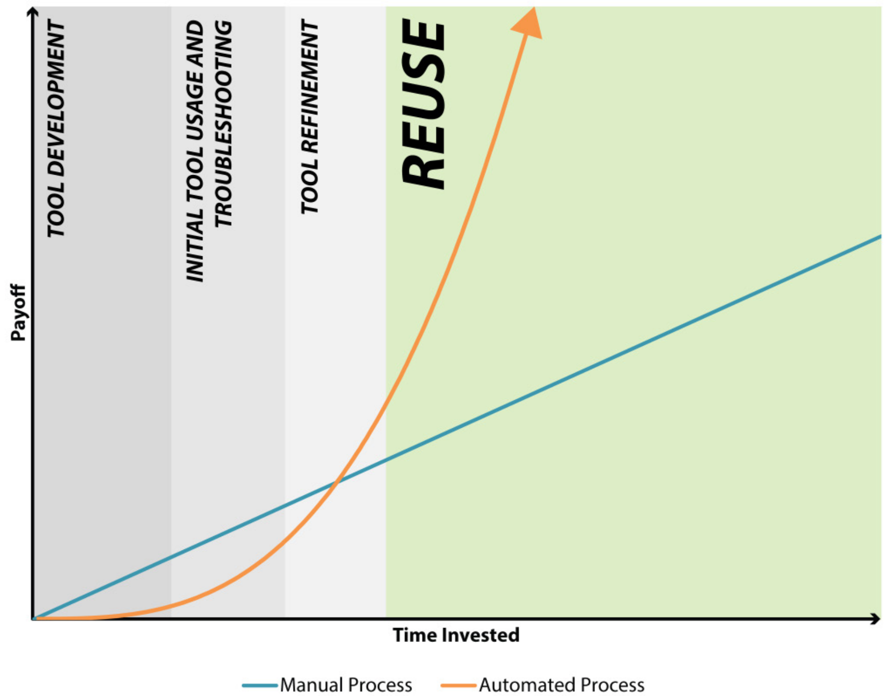

FIGURE 5: TOOLS PROVIDE MAXIMAL RETURN DURING THEIR REUSE PHASE

The Importance of Thorough Documentation and Dissemination

The value of automating processes and developing tools comes from reuse. During the reuse process, productivity is increased, time is saved, and a more reliable end product is achieved. In addition, tools allow for consistent improvement of advanced techniques. Optimization algorithms designed for reuse can be refined based on past experience and metrics. Visualization tools can be expanded or further developed to show a wider breadth of data in a more visually compelling manner, allowing the designer to accommodate changing client demands. A geometry generation script can be refined for higher levels of resolution.

The essential ingredient in tool reuse is documentation. Each tool needs an accompanying read me that clearly explains the tool's use and purpose. Code must similarly contain good commenting habits, so that developers can easily modify or troubleshoot each other's code without a lengthy "get-acquainted" period. Videos documenting the tool's previous use on projects is also invaluable, providing the blueprint for implementation while also revealing how the tool has saved time and contributed to delivering a better resolution in the past. A wide array of users can engage with complex tools with the aid of proper documentation.

Users must also locate tools in organized, standardized locations so that they are easily discovered. In addition, the development of tools should be well publicized so that users are aware of their presence. Positive dissemination habits can assure maximal return for the front end investment that tool development requires.

The Tools in Action

Examples of Integrated Modeling Environment Workflows in Action

The following images show examples of IME's included in the presentation. The first example is focused on interoperability and reflects a pre-Dynamo and early Dynamo workflow, where data is passed from one program to another in order to populate the model and documentation. Following this example are two examples of workflows where the model and its corresponding data are persistently linked between different programs, creating a truly Integrated Modeling Environment where the model may be manipulated or analyzed based on each program's strengths without a loss of data and model fidelity. The final example shows this workflow with Dynamo as the conduit, running all means of parametricism, modeling, documentation, and analysis through the Dynamo workspace.

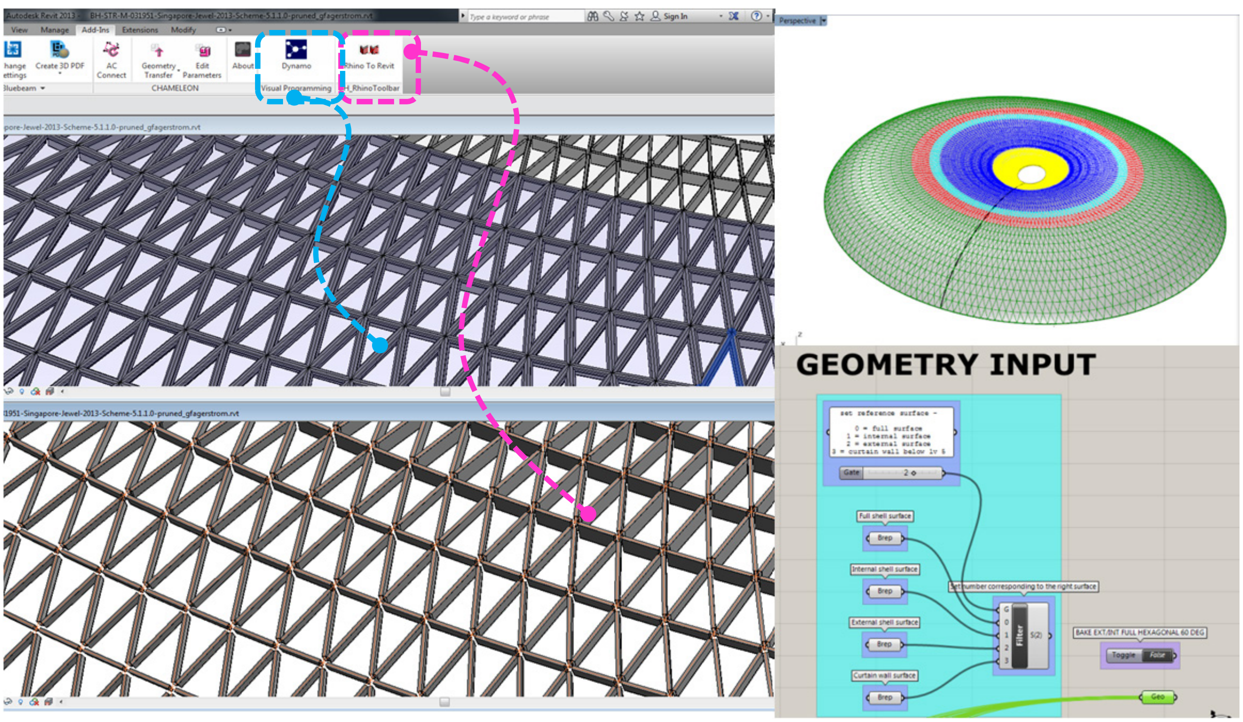

FIGURE 6: EARLY STAGE DYNAMO AND A SCRIPT-BASED PLUGIN USED FOR INTEROPERABILITY BETWEEN RHINO, GRASSHOPPER, ANALYSIS, AND REVIT (SHOWN HERE)

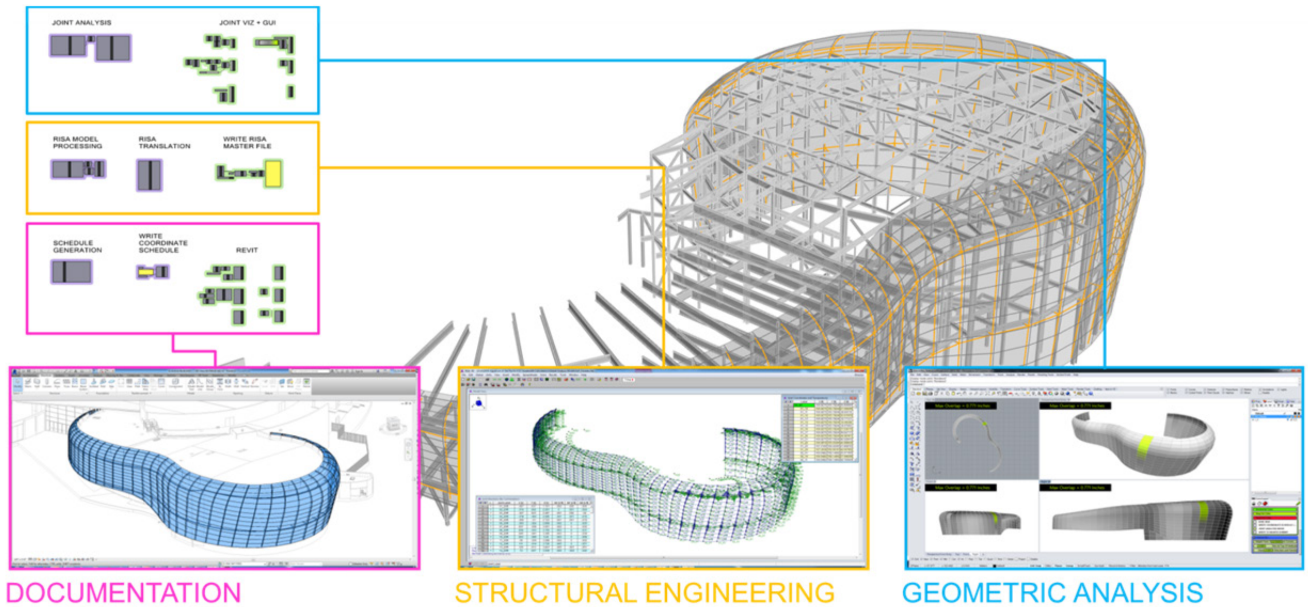

FIGURE 7: GRASSHOPPER USED IN TANDEM WITH PYTHON SCRIPTS AND DYNAMO TO SIMULTANEOUSLY ANALYZE STRUCTURAL PERFORMANCE AND GEOMETRIC PERFORMANCE WHILE ALSO MAINTAINING LIVE, PERSISTENT DOCUMENTATION AND CLASH DETECTION IN REVIT

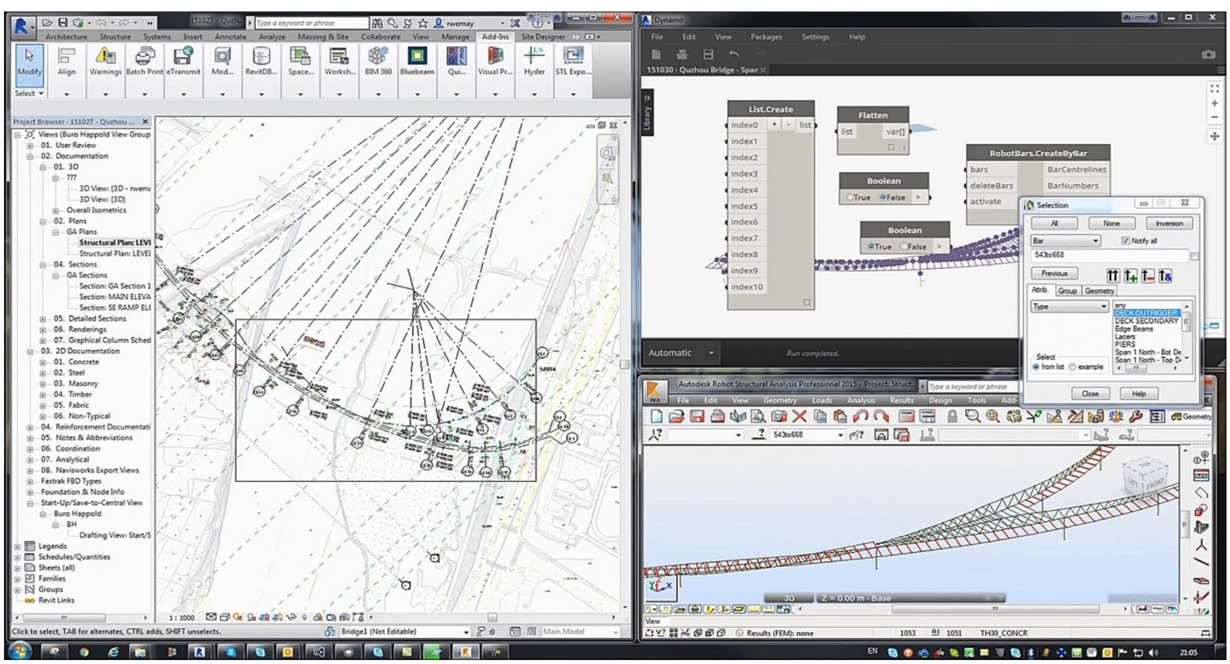

FIGURE 8: BRIDGE GEOMETRY DRIVEN BY REVIT GRIDS GENERATED PARAMETRICALLY IN DYNAMO WITH PERSISTENT LINK TO ROBOT STRUCTURAL ANALYSIS GEOMETRY