T. Fischer, K. De Biswas, J.J. Ham, R. Naka, W.X. Huang, *Beyond Codes and Pixels: Proceedings of the 17th International Conference on Computer-Aided Architectural Design Research in Asia,* 163–172.

CUSTOM DIGITAL WORKFLOWS

A new framework for design analysis integration

BIANCA TOTH
Queensland University of Technology, Australia
bianca.toth@qut.edu.au

STEFAN BOEYKENS
KU Leuven, Belgium

ANDRE CHASZAR
Delft University of Technology (TUD), Netherlands

PATRICK JANSSEN
National University of Singapore (NUS), Singapore

and

RUDI STOUFFS
TUD and NUS

Abstract. Flexible information exchange is critical to successful design integration, but current top-down, standards-based and model-oriented strategies impose restrictions that are contradictory to this flexibility. In this paper we present a bottom-up, user-controlled and process-oriented approach to linking design and analysis applications that is more responsive to the varied needs of designers and design teams. Drawing on research into scientific workflows, we present a framework for integration that capitalises on advances in cloud computing to connect discrete tools via flexible and distributed process networks. Adopting a services-oriented system architecture, we propose a web-based platform that enables data, semantics and models to be shared on the fly. We discuss potential challenges and opportunities for the development thereof as a flexible, visual, collaborative, scalable and open system.

Keywords. Visual dataflow modelling; design processes; interoperability; simulation integration; cloud-based systems.

1. Introduction

There is a clear and urgent need for better information exchange strategies to address the persistent lack of interoperability and integration in building design, analysis and construction. Recognising that most design teams use a variety of software applications and platforms, the question that remains to be answered is: How can we develop tools and technology that support designers in creating their own design processes, rather than having to adapt their processes to suit the tools' rigid requirements?

The key idea we present in this paper is that bottom-up, user-controlled and process-oriented approaches to linking design and analysis applications are more appropriate than current top-down, standards-based and model-oriented strategies, because they provide degrees of flexibility critical to the process(es) of design. This idea comes from discussions raised at the "Open Systems and Methods for Collaborative BEM (Building Environment Modelling" workshop held at the CAAD Futures 2011 Conference in Liège, Belgium, early in July 2011. Here, we continue the 'open systems' dialogue with a conceptual framework for bringing this idea into practical application, aiming to reduce current obstructions to collaborative design. We propose an open framework for integration where numerous small and specialised procedural tools are developed, adapted and linked ad-hoc, to meet the needs of individual design projects and project teams. These modular components encapsulate individual tasks that aid information exchange between domain-specific software by (semi-) automating typically tedious and non value-adding tasks associated with matching and mapping data across different schemas. A cloud-based platform enables project- and user-specific workflows to be created, shared and managed on distributed resources, via web interfaces that allow users to interact with workflows graphically. This, in combination with the elimination of file format and mapping language restrictions, ensures maximum flexibility.

Drawing on research into scientific workflows, we describe system requirements to guide future development of the proposed framework. We present a system that dispenses with an ontological premise for integration, and discuss the benefits and challenges that such a system presents for design practice and outcomes. Although no implementation of the framework has yet been created or tested, we are in the process of assembling a team of researchers and practitioners interested in pursuing our proposal. We are confident that the approach described in this framework will lend itself well to coping with the frequently changing pace and focus of design projects, as well as the varying priorities of their many stakeholders.

2. System architecture

Similar to the AEC industry, increasing complexity in scientific research and practice has led to a proliferation of specialised computational tools, each developed by different people, at different times, to support different problem-solving tasks. Across these tools, underlying data structures exhibit a high degree of heterogeneity, akin to that observed in building software. To manage this heterogeneity, and achieve the integration required to generate solutions, information must be matched and mapped across a succession of different schemas, applications and platforms (Bellahsene et al. 2011). Scientific workflows enable these information exchanges to take place quickly, reliably and flexibly, by "combining data and processes into a configurable, structured set of steps that implement semi-automated computational solutions of a scientific problem" (Altıntaş 2011, pp.9-10).

Scientific workflow systems enable the composition and execution of these complex task sequences on distributed computing resources (Deelman et al. 2009). These systems exhibit a common reference architecture, illustrated in Figure 1, and typically consist of a graphical user interface (GUI) for authoring workflows (which can also be edited textually), along with a workflow engine that handles invocation of the applications required to run the solution (Curcin and Ghanem 2008). The workflow engine supports integration between applications by engaging a combination of data-flow and control-flow constructs to handle the execution and management of tasks. Data-flow constructs establish information dependencies between tasks, and ensure that data-producing procedures are complete before data-consuming ones begin (Deelman et al. 2009). Control-flow constructs support more complex workflow interactions, such as loops and conditionals, and also coordinate the execution of tasks on remote resources (Deelman et al. 2009). Typically, control-flow constructs are overlays on the data-flow graph, either as separate nodes or layers.

Today, numerous workflow systems with different purposes and functionality exist. Some provide sophisticated interfaces and graphics, like the data visualisation application Vistrails (Callahan et al. 2006), while other more generic workflow systems, such as YAWL, are less visual but offer high-level process abstractions that can be applied to a range of usage scenarios (Curcin and Ghanem 2008). The LONI Pipeline is designed specifically to

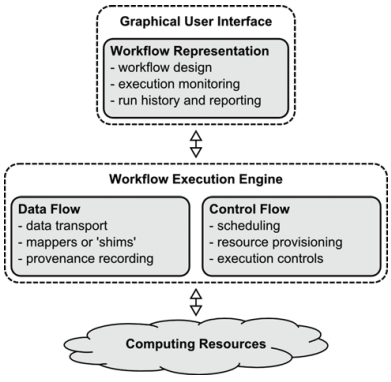

Figure 1. Workflow system architecture.

build up data processing streams for neuroimaging tools (Rex et al. 2003), while Kepler provides advanced control algorithms for actor-oriented modelling of complex physical, biological and chemical processes (Curcin and Ghanem 2008). Each system acts to accelerate and streamline the problem-solving process, however, individual capabilities vary greatly due to differences in workflow representation, data flow and control flow.

Implementation strategies for each of these three aspects of workflow are the product of specific requirements and technologies needed for the individual field or purpose for which a system is developed. In the following subsections we discuss strategies for workflow representation, data flow and control flow in relation to the needs of the AEC industry, aiming to capitalise on recent advances in cloud computing.

2.1. WORKFLOW REPRESENTATION

Workflow representation is critical for specifying tasks and dependencies. Nearly all workflow systems mentioned are visual programming tools in that they allow processes to be described graphically using some form of 'pipes-and-filters' logic. While not strictly workflow systems, programs like Grasshopper and GenerativeComponents abstract underlying CAD systems to offer similar functionality to designers for composing parametric-associative models. Each 'filter' encapsulates some data processing task, represented by a node, while a 'pipe' passes data (and in some instances control information) between two filters, represented by a connecting wire. A workflow is depicted by a network of nodes and wires to be configured and reconfigured graphically by users as required. From a user perspective, these nodes can act as a black box to perform a given function without the need for extensive or expert programming, although programming can empower the end user considerably.

Adopting this 'pipes-and-filters' architecture, our framework posits three node types: process, input/output (IO) and control. Process nodes encapsulate data analysis and transformation procedures; while the latter two node types provide functionality related to workflow initiation, execution and completion. Process nodes have a number of (typed) input and output ports for receiving and transmitting data, as well as variables that can be set by the user to guide task execution. They can be further classified into tool nodes and mapper nodes. Tool nodes wrap existing applications to make their data and functionality accessible to the workflow, while mapper nodes apply transformation procedures to data sets to map the output from one tool node to the input of another. Figure 2 shows an example network in which a Maya modelling node is connected via a series of mapper nodes (denoted by 'M') to EnergyPlus and Radiance simulation nodes. The Maya node encapsulates

a procedure that starts Maya, loads a specified model, and then generates a model instance by applying the defined parameter values. The resulting geometric output undergoes two separate transformations that map it into both EnergyPlus and Radiance compatible formats. The simulation nodes then read in this transformed data, run their respective simulations, and generate output data in the form of simulation results.

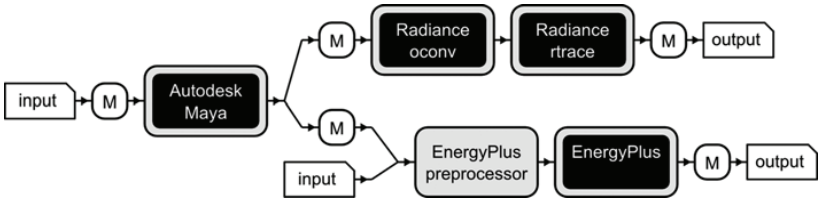

Figure 2. Exemplar workflow.

IO nodes act as data sources and sinks for the workflow. Input nodes provide the data by specifying input files and control parameters as inputs to the tool nodes and to control the data extracted from these input files. Taking the example in Figure 2, the Maya input node allows the user to specify not only the model to be used for the data source, but also particular types of geometry contained within it, while the EnergyPlus input node might simply link the appropriate weather file. Output nodes contain the workflow results, here of the EnergyPlus and Radiance simulations. They can be linked to a number of visualisation tools to display results, and users are able to define data 'mashups' in order to customise their visualisations without having to understand the coding of the underlying processes.

Control nodes apply constraints to the workflow, like conditionals and loops, which manipulate the local order of execution of nodes further along in the network. For example, an *if–then* node can force execution of different branches in a workflow, while a *repeat* node can force repeated execution of a network branch. Global control is also possible, but is defined at workflow level, rather than task level, as discussed in Section 2.3.

Users configure nodes and their dependencies using a workflow interface. Since we are describing a platform that operates in the cloud, this interface would be a web application able to access distributed cloud services. We propose a HTML5-based GUI that provides drag-and-drop functionality for placing nodes on the workflow canvas, which are then wired together by the user, similar to defining a model in Grasshopper. We also propose interface functionality to aid users in managing workflow complexity. Graphical nesting allows clusters of nodes to be collapsed into composite nodes,

facilitating modularisation of the workflow to improve its legibility (Davis et al. 2011). Provenance information retrieval and querying enables workflow history to be reviewed, so that the decision-making process can be tracked (Deelman et al. 2009).

2.2. DATA FLOW

Interoperability is a critical issue when linking applications from different domains. Scientific workflow systems deal with this in a number of ways, ranging from an ontological approach, where a common file format is imposed on data exchanges, to an open-world approach, where the user resolves data format issues as needed - a process known as 'shimming' (Altıntaş 2011). In the AEC industry, the prevailing solution to this issue is Building Information Modelling (BIM), which tends toward the all-encompassing ontological end of the spectrum. This is a top-down approach, reliant on the IFC data model and its continuous extension to cater for all possible usage scenarios. Pragmatic information exchange is assumed to evolve into process-oriented model views, where only filtered subsets of the model are exchanged (Eastman et al. 2011).

Rather than reading and writing to a common representational structure, we propose tools be coupled more effectively through procedures that allow direct data transfer, as advocated by building simulationists (Hensen et al. 2004), and transfer only needed data, rather than entire models (Augenbroe et al. 2003). While this approach is vulnerable to version changes in wrapped tools, the sharing and reuse of interoperability solutions would be a mitigating factor. Furthermore, the proposed system does not disregard BIM, but suggests IFC exchange be part of the workflow process, integrated into these custom data flows rather than forcing the whole system to adhere to its ontology. A good example of such an approach is found in the "GeometryGym" suite of tools, which enable parametric models generated in Grasshopper to be linked both to BIM workflows, through components that generate IFC objects, and to structural simulations (Mirtschin 2011).

Considering all data to be exchanged as files, we may allow any other data formats next to IFC, such as CAD data in the form of DXF, IGES or comparable proprietary formats, XML formatted data or other structured data or text, or even plain text. This list is deliberately open-ended; while the absence of any format restrictions may complicate the exchange of data amongst numerous tools, it avoids significant limitations of both an ontology-specific data representation (such as IFC) and a domain-independent general-purpose data format. Choosing the latter would undoubtedly result in situations where translations from highly customised data representations to the general-purpose format and back pose significant risks of information loss. The only necessary restric-

tion that we envision is that data formats are identified and their assumptions described, such that any mapper node may reasonably rely on these assumptions when reading in data. This restriction does not limit flexibility concerning data formats, as a new format may always be identified and described.

The selection of files as the medium for data exchange is prompted as much by the elimination of any data format restrictions as by the choice of a cloud-based platform. However, the use of distributed data files over a centralised data model may introduce data redundancy and inconsistency; e.g. when different workflow branches drawing on the same input converge, combining data from different but overlapping models. Such redundancy, admittedly, is inherent to the bottom-up approach to integration that we are advocating. Eliminating this redundancy, however, would not only greatly reduce the freedom and flexibility of designers to create their own workflow processes from any selection of tools, it would also seriously hamper the ability to define and explore unconventional design spaces.

2.3. CONTROL FLOW

As discussed in Section 2.1, control nodes provide localised ways of manipulating the workflow. To provide the desired level of flexibility, the framework also needs to offer different types of global control flow. Many existing workflow systems are restricted to simple flow mechanisms that generate topological orderings, where each node will execute only after all its predecessor nodes have executed. A key limitation of this approach, however, is that the network must be a directed acyclic graph (DAG), so networks with loops are not supported. Networks with loops, however, are clearly desirable in certain situations, such as optimisation procedures. To support loops and other node execution patterns, such as triggering nodes iteratively or periodically, different high-level control mechanisms are required. In addition to providing workflow execution functionality, these mechanisms are needed to support distributed computing, by triggering nodes to work in parallel with other nodes, as well as executing synchronously or asynchronously. To ensure maximum flexibility, the user should be able to apply different control flow mechanisms to different parts of the network. This could be achieved by assigning a control mechanism to a composite node, which would further open the possibility of nesting control flow.

3. System implementation

Implementing our framework in the cloud ensures its scalability, efficiency and reliability, as node execution can be distributed over multiple computers

in the network. Process nodes are therefore defined so that both their data and procedure can reside in the cloud. Input and output data is saved in files, and a (distributed) repository is used to manage these files. The node procedure is saved as an executable task (which may be written in any language) that reads and writes files, and a task scheduler is used to manage the execution of these tasks in the cloud.

Files resulting from one process node may be stored local to the execution of the node's task, awaiting retrieval by other process nodes. Storing files ensures a trace of all workflow output is maintained for later perusal. Files may also be copied to different locations and their copies managed within the repository. Similar to version-control systems, local file copies in combination with a local copy of the repository guarantees access to all outputs even when the user is disconnected from the cloud. File management within a repository also eliminates the need for exchanging files directly between process tasks. When a task creates output files, it registers their location (URL) together with some minimal metadata – origin, time of creation, format, etc. – for which the Dublin Core (http://dublincore.org), extended where necessary, may serve as a template. In return, it receives tokens corresponding to the various files (typically unique IDs assigned by the repository), which are passed to the task scheduler to be forwarded on to the next process node's task. The receiving task then queries the repository for the metadata and the file(s).

4. Discussion

The question of interoperability has been vexing the AEC industry for decades. The usual response is to impose standards for data formatting and construct monolithic design-analysis systems that internalise, and thus opaquely subsume, representation problems. This inflicts severe limitations on the ways in which information, and therefore designs themselves, can be described. To overcome these limitations, we are proposing that the bugaboo of BIM research and development – communication that is direct between domain-specific applications rather than via a common standard – is the preferred and even necessary arrangement. Although more work may be needed to create the multitude of possible data converters required to support such communication, this is better than over-constraint of the design process caused by the use of standardised representations and processes that are only applicable within a relatively small region of 'design space'.

Research and development of the proposed system is an ongoing effort from the Open Systems group, however, success will ultimately depend on a community of users and developers, able and willing to create, share and maintain process nodes to support various design and analysis activities. This could

potentially result in a vast collection of process nodes and an endless range of options. To aid the designer in choosing appropriately, it is important not only that a description of the functionality of each node is available, but also that the designer is able to ensure that nodes 'fit' other nodes in order to compose a valid workflow. Assertions must therefore be specified on node outputs and assumptions specified for expected inputs so that automatic checking of node compatibility is possible.

Workflow design is likely to be an incremental process in which a number of nodes are combined into a partial workflow, tested by the designer, and then further developed and extended. Besides automatic checking of the mutual fitness of adjacent nodes, the designer will need to check whether the (partial) workflow is behaving as expected and producing appropriate results. When the results are not as expected, the designer will need to debug the workflow by tracing back execution, which can be assisted by displaying intermediate, as well final, results. In the context of a vast collection of process nodes and choice of alternative ways to achieve the same or similar result, user-friendliness and knowledge-based support, the two main concerns of designers when using analysis software (Attia et al. 2009), will become crucial. Issues of accuracy, uncertainty and risk may also be of significant concern. Macdonald et al. (1999) propose the introduction of uncertainty considerations in simulations to provide meaningful feedback to the user and to improve confidence through risk assessment. While these should be addressed within individual software tools, the proposed system should also introduce this functionality into the workflow environment itself.

5. Conclusion

This paper represents an ongoing effort to address limitations in process and technology that presently obstruct design collaboration. In it we argued the need for a user-controlled and process-oriented approach to integration and interoperability, and discussed how a cloud-based workflow system can support more flexible and distributed design processes. We examined the features and functionality needed to abstract computing and data resources to make tools and technologies more accessible to users, both as individuals and as members of design teams. As well as benefiting design practice, we envisage the proposed system as a platform for researchers to share their work and increase the impact of their individual efforts through integration with other research. The system requirements that we have established will ensure that the proposed integration platform is developed to be flexible, visual, collaborative, scalable and open.

Acknowledgements

The authors would like to acknowledge and thank the participants of the "Open Systems and Methods for Collaborative BEM (Building Environment Modelling)" workshop held at the CAAD Futures 2011 Conference in Liège, Belgium, 4 July 2011, and of the LinkedIn Group sharing the same name, for their contributions to the discussions leading to the ideas presented and described in this paper. We invite interested parties to contribute to the development of these ideas and to join the discussions in the LinkedIn Group.

References

Altıntaş, İ.: 2011, *Collaborative Provenance for Workflow-driven Science and Engineering*, PhD Thesis, University of Amsterdam, Amsterdam.

Attia, S., Beltrán L., De Herde, A. and Hensen, J.: 2009, Architect friendly. A comparison of ten different building performance simulation tools, *11th IBPSA Conference,* Glasgow, 204–211.

Augenbroe, G., deWilde, P., Moon, H., Malkawi, A., Brahme, R. and Choudhary, R.: 2003, The Design Analysis Integration (DAI) initiative, *8th IBPSA Conference*, Eindhoven, 79–86.

Bellahsene, Z., Bonifati, A., Duchateau, F. and Velegrakis, Y.: 2011, On evaluating schema matching and mapping, *in* Bellahsene, Z., Bonifati, A. and Rahm, E. (eds.), *Schema Matching and Mapping*, Springer, Berlin and Heidelberg, 253–291.

Callahan, S., Freire, J., Santos, E., Scheidegger, C., Silva, C. and Vo, H.: 2006, Vistrails: visualization meets data management, *SIGMOD 2006*, Chicago, 745–747.

Curcin, V. and Ghanem, M.: 2008, Scientific workflow systems - can one size fit all? *CIBEC 2008*, Cairo, 1–9.

Davis, D., Burry, J. and Burry, M.: 2011, Untangling parametric schemata: enhancing collaboration through modular programming, *in* Leclereq, P. et al. (eds.), *Proc. 14th CAAD Futures Conference*, Liège, 55–68.

Deelman, E., Gannon, D., Shields, M. and Taylor, I.: 2009, Workflows and e-science: an overview of workflow system features and capabilities, *Future Generation Computer Systems*, **25**, 528–540.

Eastman, C., Teichholz, P., Sacks, R. and Liston, K.: 2011, *BIM Handbook - A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors*, 2nd. ed., John Wiley and Sons, Hoboken.

Hensen, J., Djunaedy, E., Radošević, M. and Yahiaoui, A.: 2004, Building performance simulation for better design: some issues and solutions, *PLEA 2004*, vol. 2, Eindhoven, 1185–1190.

MacDonald, I., Clarke, J. and Strachan, P.: 1999, Assessing uncertainty in building simulation, *6th IBPSA Conference*, vol. II, Kyoto, 683–690.

Mirtschin, J.: 2011, Engaging generative BIM workflows, *Collaborative Design of Lightweight Structures, LSAA 2011*, Sydney, 1–8.

Rex, D., Ma, J. and Toga, A.: 2003, The LONI pipeline processing environment, *Neuroimage*, **19**(3), 1033–1048.