Evo-Devo in the Sky

Patrick Janssen
National University of Singapore, Singapore
patrick@janssen.name

Abstract. *Designers interested in applying evo-devo-design methods for performance based multi-objective design exploration have typically faced two main hurdles: it's too hard and too slow. An evo-devo-design method is proposed that effectively overcomes the hurdles of skill and speed by leveraging two key technologies: computational workflows and cloud computing. In order to tackle the skills hurdle, Workflow Systems are used that allow users to define computational workflows using visual programming techniques. In order to tackle the speed hurdle, cloud computing infrastructures are used in order to allow the evolutionary process to be parallelized. We refer to the proposed method as Evo-Devo In The Sky (EDITS). This paper gives an overview of both the EDITS method and the implementation of a software environment supporting the EDITS method. Finally, a case-study is presented of the application of the EDITS method.*
Keywords. *Evolutionary algorithms; multi-objective optimisation; workflow system; cloud computing; parametric modelling.*

INTRODUCTION

Evolutionary design is loosely based on the neo-Darwinian model of evolution through natural selection (Frazer, 1995). A population of individuals is maintained and an iterative process applies a number of evolutionary procedures that create, transform, and delete individuals in the population.

Evo-devo-design differs from other types of evolutionary approaches with regards to the complexity of both the developmental procedure and the evaluation procedures. The developmental procedure generates design variants using the genes in the genotype (Kumar and Bentley, 1999). The evaluation procedures evaluate design variants with respect to certain performance metrics. These procedures will typically rely on existing stand-alone programs, including Visual Dataflow Modelling (VDM) systems and simulation programs (Janssen and Chen, 2011; Janssen et al., 2011). In many cases, these systems may be computationally expensive and slow to execute.

Designers interested in applying evo-devo-design methods for performance based multi-objective design exploration have typically faced two main hurdles: skill and speed (i.e. "it's too hard and too slow!"). From a skills perspective, the requirement for advanced interoperability engineering and software programming skills is often too demanding for designers. From the speed perspective, the requirement for processing large numbers of design variants can lead to excessively long execution times (often taking weeks to complete).

Previous research has demonstrated how these hurdles can be overcome using a VDM procedural modelling software called Sidefx Houdini (Janssen and Chen, 2011). Firstly, a number of simulation programs were embedded within this VDM system,

thereby allowing designers to define development and evaluation procedures without requiring any programming. Secondly, the evolutionary algorithm was executed using a distributed environment, thereby allowing the computational execution to be parallelized.

Although the research demonstrated how the challenges of skill and speed could be overcome, the solution was specific to the software tools being used, in particular Sidefx Houdini. Furthermore, for most designers, the proposed approach remained problematic due to the fact that they do not have access to computing grids. This paper will propose a generalized method for evo-devo-design that overcomes these limitations. The method uses two key technologies: computational workflows and cloud computing. In order to tackle the skill hurdle, computational workflow management systems are used, called Scientific Workflow Systems (Altıntaş, 2011; Deelman et al., 2008). In order to tackle the speed hurdle, readily available cloud computing infrastructure is used. We refer to the proposed method as Evo-Devo In The Sky (EDITS).

The next section will focus on the proposed EDITS method, followed by a section describing the implementation of a prototype EDITS environment. The final section will briefly present a demonstration of how the method and environment can be applied.

EDITS METHOD

An EDITS design method is proposed that overcomes the hurdles of skill and speed in a generalized way that is not linked to specific proprietary software applications.

The EDITS method enables users to run a population-based evo-devo design exploration process. This requires four computational tasks to be generated that will automatically be executed when the evolutionary process is run: initialisation, growth, feedback, and termination. The initialisation and termination tasks are executed at the start and end of the evolutionary process respectively, and perform various 'housekeeping' procedures. In addition, the initialisation task also creates the initial population of design variants.

The growth and feedback tasks are used to process design variants in the population. The growth task will take in just a single individual with a genotype and will generate a phenotype and a set of performance scores for that individual. (In the proposed method, the processes of development and evaluation are thus defined as a single growth workflow.) The feedback task will take in a pool of fully-evaluated individuals and based on a ranking of those individuals will kill some and will select some for generating new children. With just these two tasks, a huge variety of evolutionary algorithms can easily be specified. For example, if the pool size for the feedback is equal to the population size, then a generational evolutionary algorithm will result, while if pool size is much smaller than the population size, a steady-state evolutionary algorithm will result.

The first hurdle that EDITS must address is the skills hurdle. The initialisation, feedback, and termination tasks are highly standardized and can therefore be generated automatically based on a set of user-defined parameters. The growth task on the other hand is highly problem-specific and therefore needs to be defined by the user. In order to overcome the skill hurdle, the EDITS method uses a Workflow System for defining these tasks. Workflow Systems allow users to create computational procedures using a visual dataflow programming. Users are presented with a canvas for diagramming workflows as nodes and wires, where tools are represented as a nodes, and data links as wires.

Furthermore, this approach can also be used to flexibly link together existing design tools such as CAD and simulation programs. Interoperability issues can be overcome by using data mappers, whereby the output data from one tool may be linked to the input data of another tool via a set of data transform, aggregation, and compensation procedures. This approach therefore allows parametric modelling tools to be linked to simulation tools through an external coupling, which affords the user greater flexibility in tool choice and linking options.

The second hurdle to be overcome is the speed hurdle. The evolutionary process consists of a continuous process of extracting individuals from the population, processing them with the growth and feedback tasks, and inserting the updated and new individuals back into the population. Since the tasks are independent from one another, they can easily be parallelized. Cloud computing infrastructures allow users to have access to computing grids on an on-demand basis at a low cost and can therefore be used to enable such parallelization. In the proposed EDITS method, cloud computing is used for distributing the execution of both the growth and feedback tasks.

EDITS ENVIRONMENT

In order to demonstrate the EDITS method, a prototype EDITS environment has been implemented. Three key type of software are used: a distributed execution environment called Dexen, a workflow system called VisTrails, and a set of design tools, such as CAD and simulation programs.

- Dexen is a highly generic Distributed Execution Environment for running complex computational jobs on grid computing infrastructures, previously developed by the author (Janssen et al., 2011). Dexen uses a data-driven execution model, where tasks are automatically executed whenever the right type of data becomes available. Dexen consists of three main components: the Dexen Client provides a graphical user interface for managing jobs and tasks; the Dexen Server manages the population and orchestrates the execution of jobs; and Dexen Workers execute the tasks.
- VisTrails is an open-source workflow system that allows users to visually define computational workflows (Callahan et al., 2006). VisTrails uses a dataflow execution model that is well-suited to the types of procedures that need to be defined. It also provides good support for integrating existing programs. VisTrails can be used in one of two modes: in interactive mode, VisTrails provides a graphical user interface for building workflows; in batch mode, VisTrails can be used to execute previously defined workflows without requiring any user interaction.
- A set of design tools, including CAD tools (such as Houdini or Blender) and simulation tools (such as Radiance, EnergyPlus, and Calculix). (Other popular commercial CAD tools could also be integrated with this environment. However, due to inflexible licensing policies, it is currently difficult to deploy such tools in the cloud.) The CAD tools can typically run either in interactive mode or in batch mode while the simulation programs run only in batch mode, with all interaction being restricted to text based input and output files.

The EDITS environment is delivered as a cloud based service. Cloud computing can deliver services to the user at a number of different levels, ranging from delivering computing infrastructure to delivering fully functional software (Rimal et al., 2009). These levels are typically divided into three categories: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). These levels can also build on one another.

The EDITS environment is divided into three layers, corresponding to IaaS, PaaS and SaaS, as shown in Figure 1. For the base IaaS layer, the EDITS environment uses Amazon EC2, which is a commercial web service that allows users to rent virtual machines on which they run their own software. Amazon provides a web-application where users can manage their virtual machines, including starting and stopping machines. The SaaS and PaaS layers will be described in more detail below.

The SaaS layer

The SaaS layer consists of a number of graphical tools for running EDITS jobs. Overall, there are four main steps for the user: 1) starting the server, 2) creating the growth task, 3) executing the evolutionary job, and 4) reviewing progress of the job.

Step 1 involves using the Amazon EC2 web application to start an EDITS server. This simply con-

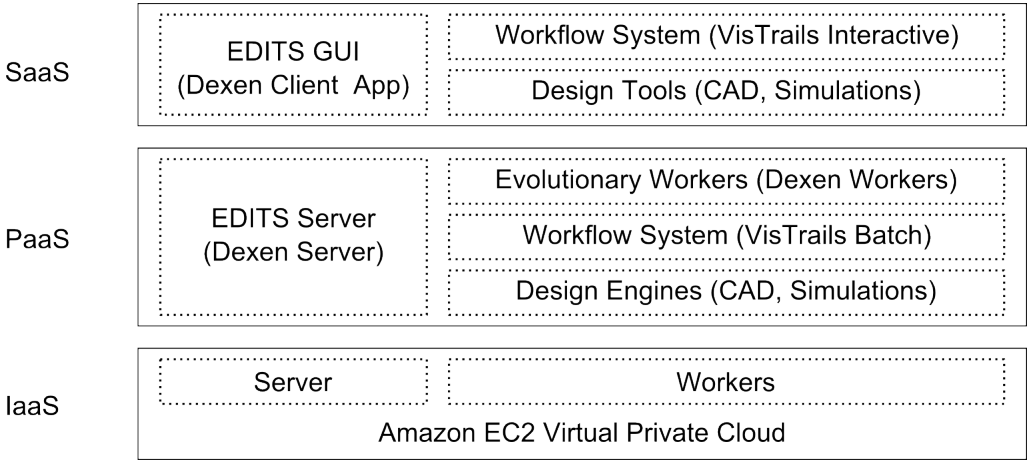


Figure 1
The three layers of the EDITS environment.

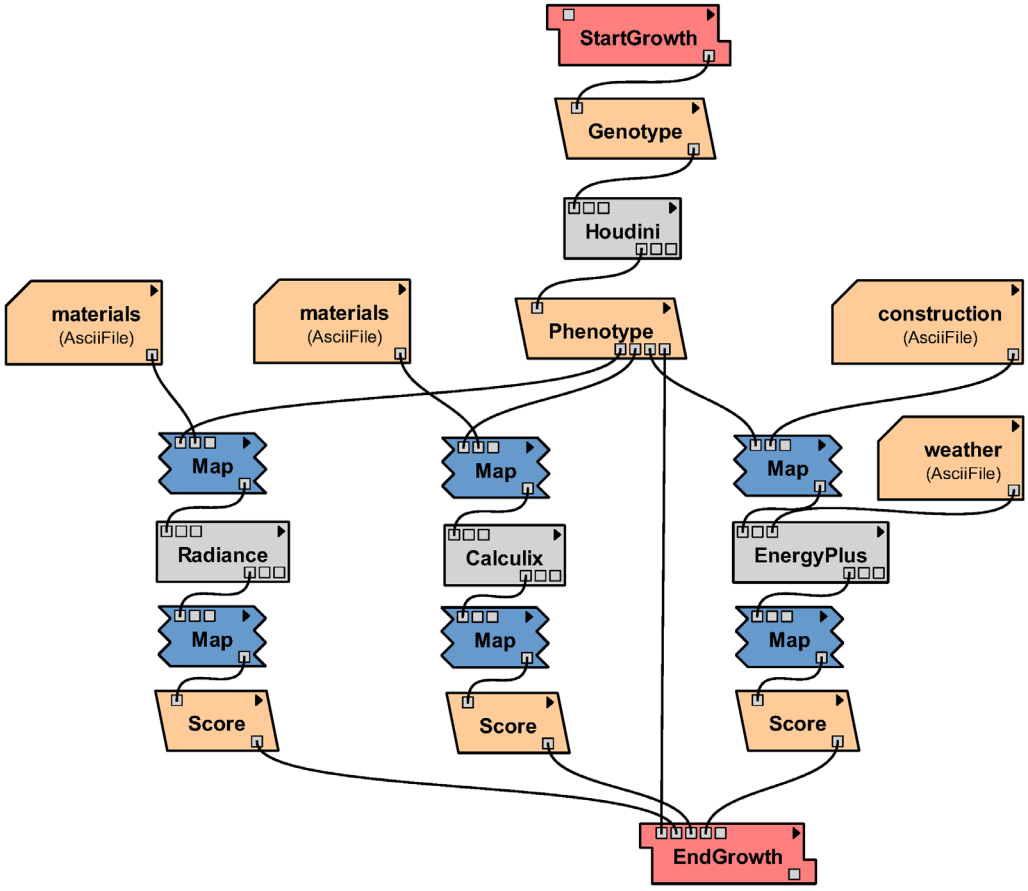
sists of logging onto the Amazon EC2 website with a standard browser, and then starting an Amazon instance. The operating system and software installed on a virtual machine is packaged as an Amazon Machine Image (AMI), and for EDITS a customized AMI has been created. This AMI is saved on the Amazon server, so it can simply be selected by the user from a list of options. The same server can be used for running multiple jobs.

In step 2, the user defines the growth task by creating a workflow with the VisTrails workflow system using a set of specially developed EDITS nodes. Figure 2 shows an example of such a workflow, consisting of a development procedure followed by three parallel evaluation procedures. The development procedure uses SideFX Houdini to generate the phenotype. The evaluation procedures use the Radiance, Calculix, and EnergyPlus simulation programs to generate performance scores. These procedures will be explained in more detail in the section describing the demonstration.

Step 3 involves executing the EDITS job. For the user, it is good if this execution could be orchestrated from within the same VisTrails environment. However, since the EDITS job may take several hours to execute, it is preferable to interact with it in an asynchronous manner. The user should be able to start the EDITS job in the cloud and then reconnect with the running EDITS job intermittently in order to download the latest results. A plugin has therefore been implemented for VisTrails that adds an EDITS menu to the menu bar for starting EDITS jobs. When a new job is started, the user can select the growth workflow, and can specify a number of parameters, including population size, mutation and crossover probabilities, selection pool size and the ranking algorithm. Once these parameters are set, a number of Python scripts required to run the job are automatically generated and uploaded to the server together with the growth workflow. The job will then start running automatically.

In step 4, the user connects to the EDITS jobs to review progress and analyse the data that is generated. Dexen has its own client application with a graphical user interface that allows users to get an overview of all the jobs that are running and to interrogate the execution of individual tasks in detail, providing information on execution time, crashes, error messages, and so forth. Data related to individual design variants can also be downloaded. However, downloading and viewing design variants one at the time can be tedious and error prone. In order to streamline this process, a set of VisTrails EDITS nodes have been created for downloading data

Figure 2
The EDITS growth workflow in the VisTrails environment.



and design variants directly from the server running in the cloud. These nodes can for example be used to create a workflow that first downloads the performance scores of all design variants and then selects a subset of these design variants for display to the user. VisTrails provides a visual spreadsheet that can be used to simultaneously display 3D models of multiple design variants (Figure 5).

The PaaS Layer

The PaaS layer builds on top of the Amazon EC2 IaaS layer, by defining an AMI for the EDITS Platform. A customised AMI was created for EDITS with all necessary software preinstalled and all settings preconfigured. The EDITS AMI includes the base operating system, together with Dexen, VisTrails, and a set of commonly used CAD and simulation programs.

The software used for orchestrating distributed execution of the EDITS job is Dexen. When the EDITS server is started on EC2, Dexen will be automatically started and all the other required software will be configured and available. The two main tasks that need to be executed are the growth and feedback tasks. Dexen maintains the population of individu-

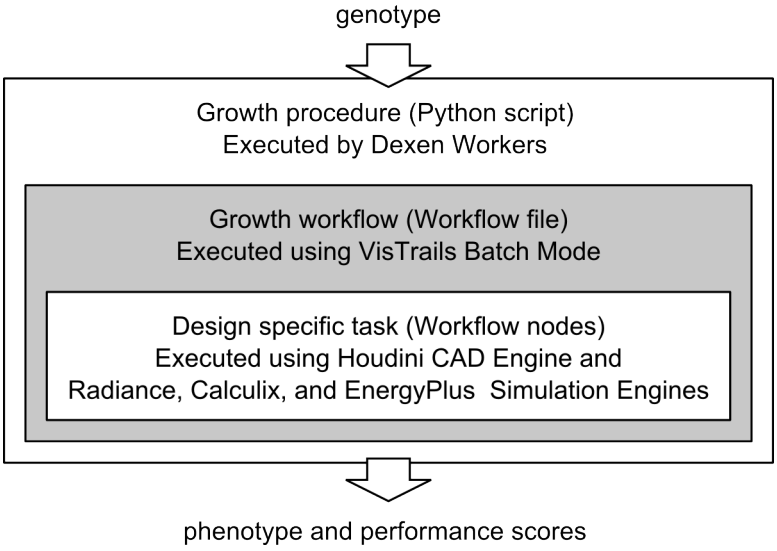


Figure 3
The software layers involved in executing the growth task. The workflow, highlighted in grey, is the only layer that needs input from the end-user.

als in a centralized database and will automatically schedule the execution of growth and feedback tasks. For the growth task, individuals are processed one at a time. For the feedback task, individuals are processed in pools of individuals, selected randomly from all fully evaluated individuals in the population. Each time either a growth or feedback task needs to be executed, Dexen will extract the individuals from the database, and send them to an available Dexen worker for processing. Once the worker has completed the task, the updated and/or new individuals will be retrieved and reinserted back into the population database.

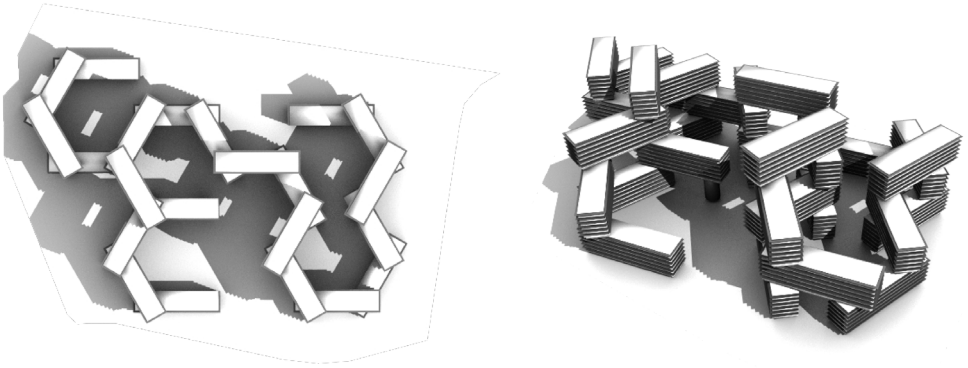
The Python scripts for the initialisation, growth, feedback, and termination tasks are automatically generated by EDITS. The growth task is the most complex due to the various layers that are involved. The task has a nested 'Russian Doll' structure, consisting of a cascade of invocations three layers deep, as shown in Figure 3. The outer layer consists of the Python script. When this script is executed, it will invoke VisTrails Batch Mode in order to execute the workflow. Since this workflow may contain numerous nodes that link to other design tools such as CAD and simulation programs, VisTrails will then invoke these design tools. For the end-user, the complexity of the growth task is hidden, since they are only required to create VisTrails workflow.

EDITS DEMONSTRATION

As a demonstration of the EDITS approach, the design for a complex residential apartment building is evolved. The case study experiment is based on the design of the Interlace by OMA. The design consists of thirty-one apartment blocks, each six stories tall. The blocks are stacked in an interlocking brick pattern, with voids between the blocks. Each stack of blocks is rotated around a set of vertical axes, thereby creating a complex interlocking configuration.

Each block is approximately 70 meters long by 16.5 meters wide, with two vertical axes of rotation spaces 45 meters apart. The axes of rotation coincide with the location of the vertical cores of the building, thereby allowing for a single vertical core to connect blocks at different levels. The blocks are almost totally glazed, with large windows on all four facades. In addition, blocks also have a series of balconies, both projecting out from the facade and in-

Figure 4
The initial configuration based on the original design, consisting of 31 blocks in 22 stacks of varying heights.



set into the facade. The initial configuration, shown in Figure 4, is based on the original design by OMA. The blocks are arranged into 22 stacks of varying height, and the stacks are then rotated into a hexagonal pattern constrained within the site boundaries. At the highest point, the blocks are stacked four high.

For the case study, new configurations of these 31 blocks were sought that optimise certain performance measures. For the new configurations, the size and number of blocks will remain the same, but the way that they are stacked and rotated can differ. A VisTrails growth workflow was defined that performed both development and three evaluations. The workflow shown in Figure 2 was developed for this demonstration.

Growth workflow: design development

For the procedural modelling of phenotypes, SideFX Houdini was used. For the genotype to phenotype mapping, an encoding technique was developed called *decision chain encoding* (Janssen and Kaushik, 2013). At each decision point in the modelling process, a set of rules is used to generate, filter, and select valid options for the next stage of the modelling process. The generate step uses the rules to create a set of options. The filter step discards invalid options that contravene constraints. The select step chooses one of the valid options. In order to minimise the complexity of the modelling process, options are generated in skeletal form with a minimum amount of detail. The full detailed model is then generated only at the end, once the decision chain has finished completing.

In the decision chain encoding process, the placement of each of the 31 blocks is defined as a decision point. The process places one block at the time, starting with the first block on the empty site. At each decision point, a set of rules is used to generate, filter, and select possible positions for the next block. Each genotype has 32 genes, and all are real values in the range {0,1}. In the generation step, possible positions for the next block will be created using a few simple rules. First, locations are identified, and second orientations for each location are identified. The locations are always defined relative to the existing blocks already placed, and could be either on top of or underneath those blocks. The orientations are then generated in 15° increments in a 180° sweep perpendicular to either end of the existing block. In the filtering step, constraints relating to proximity between blocks and proximity to the site boundary are applied, thereby ensuring that only the valid positions remain. In the selection step, the decision gene in the genotype chooses one of the valid block positions.

The resulting phenotypes consist of simple polygonal models. Three separate files are generated, one for each of the simulations. These models represent different sub-sets of information relating to

the same design variant. These sub-sets of information are selected in order to match the data requirements of the simulation programs. In order to facilitate the data mapping, custom attributes are defined for geometric elements in the model. For example, polygons may have attributes that define key characteristics, such as *block* (e.g. block1, block2), *type* (e.g. wall, floor, ceiling), and *parent* (e.g. the parent of the shade is the window; the parent of the window is the wall). These attributes are used by the mapping nodes in order to generate appropriate input files for the simulations. The geometry together with the attributes are saved as JSON files (i.e. simple text files).

Growth workflow: design evaluations

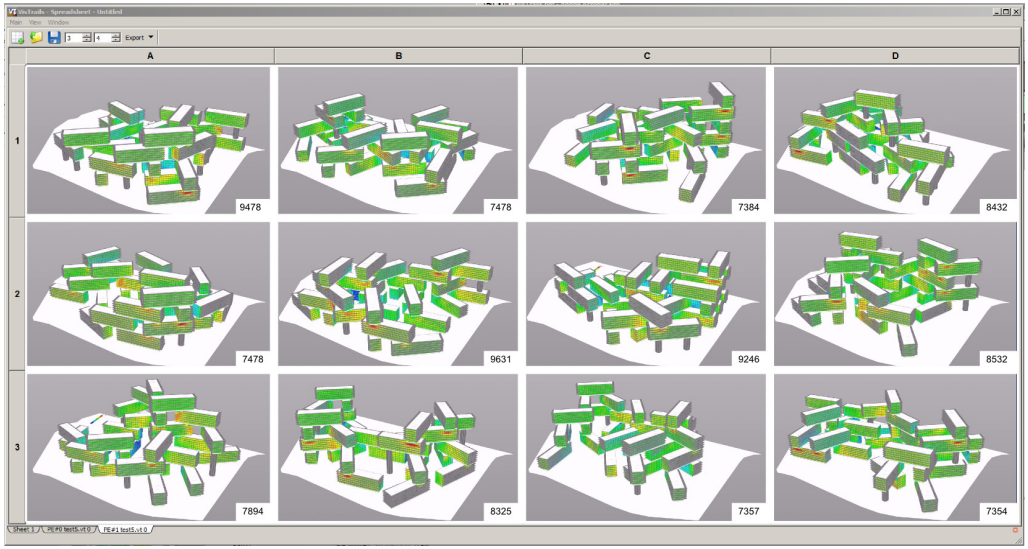
For the multi-objective evaluation, three performance criteria were defined: maximisation of daylight, minimisation of structural strain, and minimisation of cooling load. These performance criteria have been selected in order to explore possible conflicts. For example, if the blocks are clustered close together the cooling load will decrease due to inter-block shading but the daylight levels will also reduce. If the blocks are stacked higher, then they are likely to get better daylight but they may become less structurally stable. The three performance criteria are calculated as follows:

- Maximisation of daylight: An evaluation is defined that executes Radiance in order to calculate daylight levels on all windows under a cloudy overcast sky. The amount of light entering each window is then adjusted according to the visual transmittance of the glazing system for that window. The performance criterion is defined as the maximization of the total number of windows where the light entering the window is above a certain threshold level for reasonable visual comfort.
- Minimisation of structural strain: An evaluation is defined that executes Calculix in order to calculate the global structural behaviour using Finite Element Analysis (FEA) under various loading conditions. In order to reduce the computational complexity, the building configuration is modelled in a simplified way, by grouping individual structural elements into larger wholes called *super-elements* (Guyan, 1965). The performance criterion is defined as the minimisation of the maximum strain within the structure.
- Minimisation of cooling load: An evaluation is defined that executes EnergyPlus in order to calculate the cooling load required in order to maintain interior temperatures below a certain threshold for a typical schedule. In order to reduce the computational complexity, an ideal-load air system together with a simplified zoning model is used, and the simulation is run for a periods of one week at the solstices and equinoxes. The performance criterion is defined as the minimisation of the average daily cooling load.

In Figure 2, the three workflow branches defining the evaluation procedures are shown. Each evaluation procedure includes two mapper nodes: an input mapper for generating the required input files, and an output mapper for generating the final performance score. These mapper nodes are currently implemented as Python scripts, but part of this research is the development of a graphical application for defining mapper nodes. See Janssen at al. (2013) for more details.

The input mappers transform the JSON files from the developmental procedure to the appropriate input files for the simulations. As well as the geometry information from these JSON files, the mappers also require other material information. The output mappers transform the raw simulation data into performance scores: for the Radiance data, the mapper calculates the number of windows below the daylight threshold; for Calculix, the mapper calculates the maximum strain in the structure; and, for EnergyPlus, the mapper calculates the average daily cooling load. These three evaluation scores are then provided as the final output of the growth task.

Figure 5
A set of design variants shown in the visual spreadsheet tool within VisTrails.



Results

When running the job, the population size was set to 200 and a simple asynchronous steady-state evolutionary algorithm was used. Each generation, 50 individuals were randomly selected from the population and ranked using multi-objective Pareto ranking. The two design variants with the lowest rank were killed, and the two design variants with the highest rank (rank 1) were used as parents for reproduction. Standard crossover and mutation operators for real-valued genotypes were used, with a mutation probability being set to 0.01. Reproduction between pairs of parents results in two new children, thereby ensuring that the population size remains constant.

The evolutionary algorithm was run for a total of 10,000 births, taking approximately 8 hours to execute. The final non-dominated Pareto set for the whole population contained a range of design variants with differing performance tradeoffs.

A workflow was created in order to retrieve and display designs from the Pareto front. A selection of design variants are shown in Figure 5.

CONCLUSIONS

For designers, the EDITS approach allows two key hurdles of skills and speed to be overcome. First, it overcomes the skills hurdle by allowing designer to define growth tasks as workflows using visual programming techniques. Second, it overcomes the speed hurdle by using cloud computing infrastructures to parallelize the evolutionary process. The demonstration case-study shows how the EDITS approach can be applied to a complex design scenario.

Future research will focus on the development of VisTrails data analytics nodes. This would allow users to create workflows to perform various types of analysis on the data generated by the evolutionary process, including hypervolume and clustering analysis.

REFERENCES

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

Callahan, S, Freire, J, Santos, E, Scheidegger, C, Silva, C and Vo, H 2006, 'Vistrails: Visualization Meets Data Man-

agement', *Proceedings of the SIGMOD*, Chicago, pp. 745–747.

Deelman, E, Gannon, D, Shields, M and Taylor, I 2008, 'Workflows and e-Science: An Overview of Workflow System Features and Capabilities', *Future Generation Computer Systems*, pp. 528–540.

Frazer, JH 1995, *An Evolutionary Architecture*, AA Publications, London, UK.

Guyan, RJ 1965, 'Reduction of Stiffness and Mass Matrices', *AIAA Journal,* 3(2), pp. 380–380.

Janssen, PHT, Basol, C and Chen, KW 2011, 'Evolutionary Developmental Design for Non-Programmers', *Proceedings of the eCAADe Conference*, Ljubljana, Slovenia, pp. 886–894.

Janssen, PHT, Chen, KW and Basol, C 2011, 'Iterative Virtual Prototyping: Performance Based Design Exploration', *Proceedings of the eCAADe Conference*, Ljubljana, Slovenia, pp. 253–260.

Janssen, PHT and Chen, KW 2011, 'Visual Dataflow Modelling: A Comparison of Three Systems', *Proceedings of the CAAD Futures Conference*, Liege, Belgium, pp. 801–816.

Janssen, PHT and Kaushik, V 2012, 'Iterative Refinement Through Simulation: Exploring Trade-off s Between Speed and Accuracy', *Proceedings of the 30th eCAADe Conference*, pp. 555–563.

Janssen, PHT and Kaushik, V 2013, 'Decision Chain Encoding: Evolutionary Design Optimization with Complex Constraints', *Proceedings of the 2nd EvoMUSART Conference*, pp. 157–167.

Janssen, PHT, Stouffs, R, Chaszar, A, Boeykens S and Toth B, 'Data Transformations in Custom Digital Workflows: Property Graphs as a Data Model for User-Defined Mappings', *Intelligent Computing in Engineering Conference - ICE2012*, pp. 1–10.

Kumar, S and Bentley, PJ 1999, 'The ABCs of Evolutionary Design: Investigating the Evolvability of Embryogenies for Morphogenesis', *Proceedings of the GECCO*, pp. 164–170.

Rimal, BP, Eunmi, C and Lumb, I 2009, 'A Taxonomy and Survey of Cloud Computing Systems', *Proceedings of the INC, IMS and IDC Joint Conference*, pp. 25–27.