ELSEVIER

Automation in Construction 14 (2005) 253–264

AUTOMATION IN CONSTRUCTION

www.elsevier.com/locate/autcon

Towards integrated performance-driven generative design tools

Kristina Shea[a,*], Robert Aish[b], Marina Gourtovaia[a]

[a]*Engineering Design Centre, Cambridge University Engineering Department, Trumpington Street, Cambridge CB2 1PZ, UK*
[b]*Bentley Systems, Exton, PA, USA*

Abstract

Performance-driven generative design methods are capable of producing concepts and stimulating solutions based on robust and rigorous models of design conditions and performance criteria. Using generative methods, the computer becomes a design generator in addition to its more conventional role as draftsperson, visualizor, data checker and performance analyst. To enable designers to readily develop meaningful input models, this paper describes a preliminary integration of a generative structural design system, *eifForm*, and an associative modeling system, *Generative Components*, through the use of XML models. An example is given involving generation of 20 lightweight, cantilever roof trusses for a saddle shaped stadium roof modeled in *Generative Components*. Synergies between the two systems and future extensions are discussed.

Keywords: Generative design; Parametric and associative geometry; Design optimization; Performance-based design; Computer aided design tools

1. Introduction

The convergence of the geometric freedom offered by current CAD tools and new manufacturing capabilities provide great opportunities for designers and architects to move away from traditional compositions based on symmetry and repetition in order to explore new innovative forms. Digital design tools have now moved way beyond their early use, which was generally limited to the production of final drawings. An integrated design process involving CAD, solid modeling, a range of performance analysis tools, and rapid prototyping is emerging and has been essential in the design of today's most unique structures and buildings. With the recent development of parametric and associative geometry, CAD tools are now able to parametrically vary design concepts in step with designer intent. However, the fundamental geometric constructions remain static. The next phase in digital design lies in considering the computer as a collaborative partner in the design process that is capable of generating ideas and stimulating solutions in response to robust and rigorous models of design conditions and performance. Incorporating performance models to guide computational generation

* Corresponding author. Tel.: +44 1223 332 666; fax: +44 1233 332 662.
E-mail address: ks273@cam.ac.uk (K. Shea).

0926-5805/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.autcon.2004.07.002

process will yield tools that help architects, designers and engineers think critically about system, rather than just component, performance from different viewpoints, e.g. engineering performance, spatial performance, cost, and fabrication, throughout design conception of innovative forms. However, providing tools that enable designers to readily develop these robust and rigorous input models that describe their design intent in order to guide design generation remains a challenge.

This paper gives an overview of combining a generative structural design system, *eifForm*, with an associative modeling system, *Generative Components* (formerly called *Custom Objects*). The initial integration described here gives an opportunity to explore the complexity involved in combining such tools, the synergies between associative geometry and performance-driven generative design, as well as their combined potential for enhancing negotiation between architects and engineers in the development of novel yet efficient and buildable forms. A new way of designing architectural form and associated structure in parallel is emerging.

Parametric modeling as a concept and mathematical construct, e.g. parametric curves and surfaces, has been around for years with the first parametric CAD tools emerging in 1989. Parametric modeling involves the use of geometric constraints as well as dimensional relations and data to drive shape definition [2,16]. Values within parametric expressions can be modified by designers and are then propagated through a design, i.e. strategic manipulation. Alternatively, associative geometry allows dynamic manipulation of user-defined dependency relations by graphical manipulation, i.e. intuitive manipulation.

Often building on parametric concepts, generative design transforms the computer from a modeling assistant to a generator through the use of a defined set of production, or grammar, "rules". Generative systems are aimed at both sparking new design ideas and solving difficult tasks, both of which provide design assistance and extend designers' current capabilities. In architectural design, shape grammars have been utilized for generating new spatial designs and characterizing designer style [8], such as Siza's housing types [5]. Shape grammars [14] can be basic grammars where rules are described such that they are applicable to, say, any square only of a certain dimension, or parametric grammars, where rules are applicable to, say, any four-sided polygon, e.g. squares, rectangles, rhombi, depending on the allowable geometric transformations. While a significant amount of theory has been developed for shape grammars [8], few implemented systems in architecture encode 3D parametric grammars; one example can be found in Ref. [6]. In addition, few implemented generative systems, in general, incorporate physical performance feedback. An example can be found in Ref. [3] where a genetic algorithm is combined with lighting and thermal analysis to generate novel and performance driven building envelopes and room configurations.

2. Generative Components and eifForm

Generative Components is a graph-based associative geometry modeling system that combines geometric modeling and programming [1]. The project aims to provide a set of common parametric tools within *Microstation* and *Triforma*, the current CAD tools offered by Bentley Systems. While it includes capabilities common to other parametric modeling tools, it also provides for intuitive, graphical manipulation of models and for reprogramming the tool. The latter is done through the creation of "feature classes" and is aimed at making more customized and responsive models.

The generative method within *eifForm* is an optimizing process called structural topology and shape annealing (STSA), which combines structural grammars, performance evaluation including structural analysis and performance metrics, and stochastic optimization via simulated annealing [7,15]. An overview of the main method steps within the iterative process is shown in Fig. 1 and a sample screenshot of *eifForm* in Fig. 2. The optimization technique used does not guarantee that exact mathematically optimal solutions are generated, but rather supports optimally directed design exploration. This enables incorporation of a structural grammar that generates new structural members throughout the optimization process, a wide range of real, complex design constraints and generation of multiple alternative designs from a single starting point. The method thus supports optimally directed exploration of lattice-based struc-

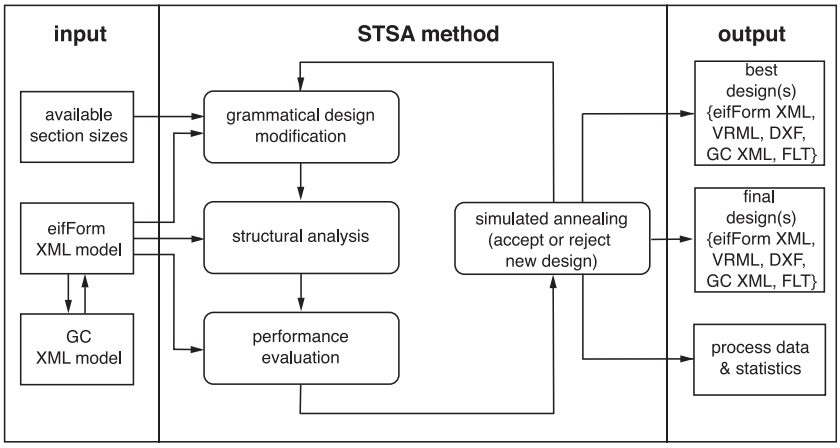

Fig. 1. Structural topology and shape annealing (STSA) method overview.

tures in relation to physical behavior, spatial and cost performance for both routine and challenging scenarios. Compared to published results in structural optimization, the method is capable of generating efficient and innovative planar trusses [9,10], single-layer space trusses [11] and full-scale transmission towers [12].

While the resulting designs produced by the method have gained great interest, a main challenge in integrating the system within a design process remains enabling designers to develop an understanding of how to create the necessary input models such that they best reflect the requirements and conditions of a design task at hand. Using *eifForm* effectively requires designers to create initial geometric, generative and performance models that describe design intent. The geometric model includes a description of the geometry and topology, or structural connectivity, of a starting design, e.g. points, lines, and shapes. The generative model includes descriptions of generative parameters, e.g. the maximum number of structural members in a candidate design, parametric constraints on the structural grammar, and parameters that control the

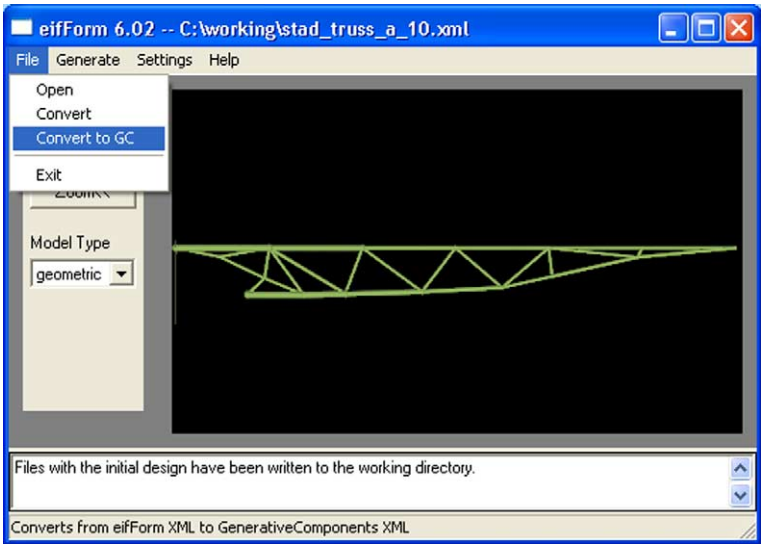

Fig. 2. Main GUI window of *eifForm*.

generative and optimization process. The performance model includes structural conditions, such as material type and loading, and an optimization model, that is a description of design objectives and performance requirements. Necessary input models can be created by the user in a text editor either from scratch, possibly using geometry output from a CAD system, or by modifying an existing valid *eifForm* input file.

The motivation for integrating *eifForm* and *Generative Components* is to enable:

- the development of richer input models (currently geometry alone) in *Generative Components* for transfer to *eifForm*,
- effective visualization and manipulation of the geometric and topologic relations that are fundamental to the method within *eifForm*,
- improved exploration of novel structural designs by the ability to place generated designs of structural systems back within an integrated parametric model of the whole design, which can be easily modified to investigate 'what-if' scenarios, and
- exploration of shape modifications in an associative geometry environment before transfer to *eifForm* that will then automatically generate the many, often combinatorially explosive, design possibilities they may produce.

The first two points will be highlighted in this paper and the potential for the second two points will be discussed.

3. Integration via eXtensible Markup Language (XML) models

A 'federated' systems architecture is used to relate the two tools, with eXtensible Markup Language (XML) employed for their preliminary integration. XML has become a language of choice for enabling integration of proprietary applications [4]. XML provides flexible and adaptable information identification since it is a meta-language, i.e. a language for describing other languages, which allows one to design their own customized markup languages for most any type of document. Flexibility and interoperability make XML a favorite data format for two-dimensional web-graphics, archiving, encoding geotechnical information, 3D modeling and VRML rendering, as well as attracting growing interest in the AEC community. Other factors contributing to rapid growth in use of XML is that it is an open standard, has a good specification, is human-readable, is intuitive and flexible, can carry almost any type of information and off-the-shelf parsers are widely available for most common programming languages today, e.g. C, C+, C#, Java, Perl and Python.

In this integration, XML models of design scenarios, e.g. a set of 20 stadium roof trusses, are created in *Generative Components* and then imported into *eifForm* to create a starting point for design generation. After a set of optimized structural designs are generated, *eifForm* then exports new XML models back directly into the original design context modeled within *Generative Components*. The exchange of information between tools requires more than just transfer of geometric data since a design scenario within *eifForm* is modeled in terms of both geometric and topologic attributes of an initial design. XML has proven to be very beneficial for this purpose for the reasons given previously, even though there is currently no AEC standard for XML structural models.

XML does not provide a model for data, rather, it reflects an application's own data model. In the absence of a standard, each system has its own XML model for essentially the same data. However, provided that systems document their XML formats well, writing converters between different XML formats is straightforward. This is the current level of integration here. One can envisage that a tool that converts automatically between different arbitrary XML documents with minimal pre-configuration by the user might appear soon.

The XML model file for *eifForm* (Fig. 3) consists, as it should be for any XML document, from one top-level document element, called *eifForm*, which encapsulates a number of nested elements. As *eifForm*'s input data has a hybrid character, four types of children elements of the *eifForm* element are used, namely, Settings, StructuralProperties, Obstacles and Design.

The Settings element lists the names of settings and their values that users can modify to influence the design generation process, i.e. the generative model

```
<EifForm Version="B6.02" DesignName="stadium_truss_a0.xml">
  <Settings
       min-length="267.0" max-length="3000.0" min-angle="10.0"
       max-lines="50" min-shapes ="2" max-joint-members="8"
       guage="true" guage-path="CHS_sizes_114" gravity="9.81"
       max-displacement="0.0349" mass-weight="1.0" >
    <buckling-constraint Value="true" /> <stress-constraint Value="true" />
    <displacement-constraint Value="true" /> </Settings>
  <StructuralProperties>
    <LoadCases TotalNumber="1">
      <LoadCase Index="1"> <LoadCaseLoad Type="point" />
                           <LoadCaseLoad Type="self-weight" /> </LoadCase>
    <Loads TotalNumber="1">
      <Load Index="1" ForceX="0.0" ForceY="476317.0" ForceZ="0.0"
         MomentX="0.0" MomentY="0.0" MomentZ="0.0"/> </Loads>
    <Materials TotalNumber="1">
      <Material Index="1" Name="Steel" YoungsModulus="20670000.0"
          Density="0.00785" MaxCompressiveStress="20400.0"
          MaxTensileStress="20400.0"/> </Materials>
  </StructuralProperties>
  <Obstacles TotalNumber="1">
    <Obstacle Index="1" MinX="0.0" MinY="610.0" MinZ="0.0"
       MaxX="7290.0" MaxY="10000.0" MaxZ="0.0"/> </Obstacles>
  <Design Type="planar">
    <CS LocalOrigin="{-2.109424E-13, 11584.016450, 4165.242043}"
       Xvector="{2.030753E-17, -0.993788, 0.111289}"
       Yvector="{-1.225675E-16, -0.111289, -0.993788}"
       Zvector="{1.000000, 6.540969E-18, -1.240661E-16}" />
    <Sections TotalNumber="1"> <Section Index="1" Area="50.0"/> </Sections>
    <Points TotalNumber="3">
      <Point Index="1" X="0.000" Y="0.000" Z="0.000" Permanent="true"
         LoadIndex="none"> <BoundaryConditions X="true" Y="true" Z="true"/>
         <MoveConditions X="false" Y="false" Z="false"/> </Point>
      <Point Index="2" X="7289.403" Y="-0.014" Z="1.850E-13" Permanent="true"
         LoadIndex="1 (LoadCase 1)"> <BoundaryConditions X="false" Y="false"
         Z="true"/> <MoveConditions X="false" Y="false" Z="false"/> </Point>
      <Point Index="3" X="935.237" Y="607.853" Z="7.889E-29" Permanent="true"
         LoadIndex="none"> <BoundaryConditions X="true" Y="true" Z="true"/>
         <MoveConditions X="false" Y="false" Z="false"/> </Point>
    </Points>
    <Lines TotalNumber="3">
      <Line Index="1" Start="1" End="2" SectionIndex="1" MaterialIndex="1"/>
      <Line Index="2" Start="2" End="3" SectionIndex="1" MaterialIndex="1"/>
      <Line Index="3" Start="3" End="1" SectionIndex="1" MaterialIndex="1"/>
    </Lines>
    <Shapes TotalNumber="1">
      <Shape Index="1" FirstPoint="1" SecondPoint="2" ThirdPoint="3"/>
    </Shapes>
  </Design>
</EifForm>
```

Fig. 3. An abridged *eifForm* XML file describing an initial design for a single stadium roof cantilever truss.

and performance model. A manual for the program gives a full listing of possible settings, currently 60 in total, and guidance on how to steer the generative process in a desired and required direction by a meaningful choice of settings and their values. Settings are (**name**, **value**) pairs, where **name** is the name of one of the global variables in the program and **value** is the value that is assigned to the variable. Most settings are the attributes of the Settings element and are listed as a **name**="**value**" pair. A limited number of settings have a more complex nature and are represented by child elements within the Settings element, e.g. the buckling-constraint setting.

The StructuralProperties element defines the structural model used in the structural analysis and lists the material properties, a range of applied loads and load cases. The Obstacles element defines 3D geometric obstacles that encourage generation of structural members around, but not within, these spaces. The Design element describes the geometry of the scenario, i.e. Points and Lines including references to the StructuralProperties element, as well as the topology of a design, i.e. Shapes that each contain an ordered list of references to Points. A Design also contains a description of a local coordinate system (CS) defined in *Generative Components*. Together the four elements (Settings, StructuralProperties, Obstacles, Design) establish a relation between the geometry, topology, structural model, generative model and performance model settings.

The XML model in *Generative Components* contains similar information as in the Design element

```
<CustomObjectsScript>
   <Transaction Text="create default base_cs">
     <base_cs Model="&amp;quot;Default&amp;quot;"
     Type="CustomObjects.UserFeatures.CoordinateSystem"/>
   </Transaction>
   <Transaction Text="create eifform_cs">
     <eifform_cs LocalOrigin="{-2.109424E-13, 11584.016450, 4165.242043}"
       SubModel="base_cs"
       Type="CustomObjects.UserFeatures.CoordinateSystem"
       Xvector="{2.030753E-17, -0.993788, 0.111289}"
       Yvector="{-1.225675E-16, -0.111289, -0.993788}"
       Zvector="{1.000000, 6.540969E-18, -1.240661E-16}"/>
   </Transaction>
   <Transaction Text="points">
     <point_1 SubModel="eifform_cs" Type="CustomObjects.UserFeatures.Point"
        XYZlocal="{0.000, 0.0,00 0.000}"/>
     <point_2 SubModel="eifform_cs" Type="CustomObjects.UserFeatures.Point"
        XYZlocal="{7289.403, -0.014, 1.850E-13}"/>
     <point_3 SubModel="eifform_cs" Type="CustomObjects.UserFeatures.Point"
        XYZlocal="{935.237, 607.853, 7.889E-29}"/>
   </Transaction>
   <Transaction Text="lines">
     <cone_1 Color="7" Radius="7.75" StartPoint="point_1" EndPoint="point_2"
       Type="CustomObjects.UserFeatures.Cone"/>
     <cone_2 Color="7" Radius="7.75" StartPoint="point_2" EndPoint="point_3"
       Type="CustomObjects.UserFeatures.Cone"/>
     <cone_3 Color="7" Radius="7.75" StartPoint="point_3" EndPoint="point_1"
       Type="CustomObjects.UserFeatures.Cone"/>
   </Transaction>
   <Transaction Text="shapes">
     <shape_1 Color="253" Fill="true" Type="CustomObjects.UserFeatures.Shape"
       Vertices="{point_1, point_2, point_3, point_1}"/>
   </Transaction>
</CustomObjectsScript>
```

Fig. 4. An abridged *Generative Components* XML model describing the same initial design for a single stadium roof cantilever truss as in Fig. 3.

of the *eifForm* XML model, but in a different format (Fig. 4). Here, XML serves as a script file executed or "played" within *Generative Components*. Each action is described as a transaction, rather than the more conventional use in *eifForm* as an input file. Topologic description of a design is encoded through the use of the Shape command, which, similar to the use in *eifForm*, defines an ordered list of references to

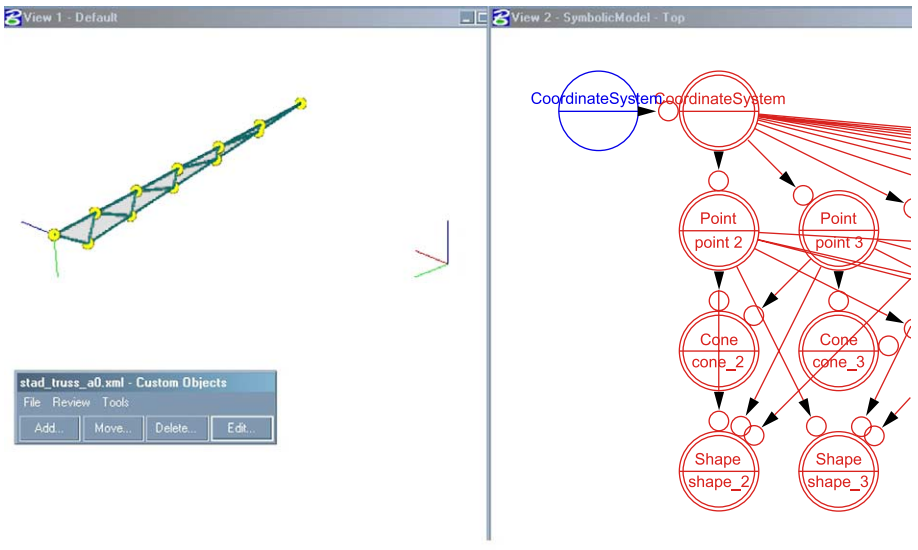

Fig. 5. A sample initial cantilever truss model in *Generative Components* and corresponding symbolic graph illustrating the necessary *eifForm* input, namely points, lines (described in 3D as cones) and shapes (shaded left for illustration only). Each cantilever truss within the stadium roof system is attached to its own local coordinate system.

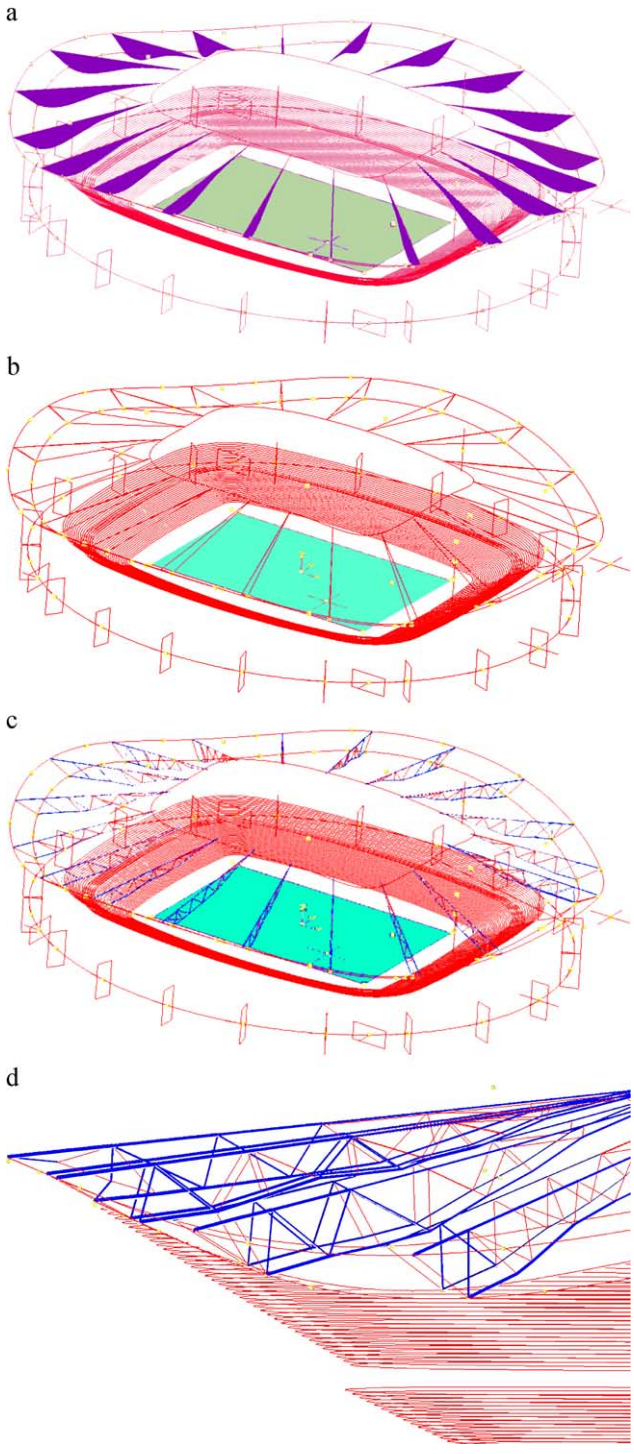

Fig. 6. A set of 20 optimized cantilever trusses with seven unique spans generated by *eifForm* for a stadium roof model created in *Generative Components* (GC): (a) initial design illustration in GC, (b) minimum initial designs for input to *eifForm*, (c) a set of generated cantilevers visualized within the stadium concept, (d) a close-up view.

points. In *Generative Components* each transaction is executed sequentially producing an interlinked symbolic model and 3D graphical model (Fig. 5). Once the model has been executed in *Generative Components*, it can be reviewed and modified and then transferred back to *eifForm*.

4. An example: a stadium roof truss system

To demonstrate the preliminary integration described and explore future potential, *eifForm* is used to generate and optimize a structural roof truss system consisting of 20 cantilever trusses with seven unique spans for a saddle-shaped stadium roof defined by a parametric model of a whole stadium concept in *Generative Components*; see Fig. 6a. Each initial truss structure includes a starting shape defined by its location in the stadium roof, which is described by a reference to a unique local coordinate system in *Generative Components*. The cantilevers span from 58.68 to 72.89 m, moving around the roof shape, and the height of the support points along the angled rim of the stadium is constant for all spans at 6.08 m (Fig. 6b). The initial design for *eifForm* can be either just a single triangle between the support points and cantilever tip (Fig. 6b) or an initial lacing produced in *Generative Components* (Fig. 5). The trusses are made of steel (S355) circular hollow sections (CHS) and are required to withstand a distributed loading across the top chord of each cantilever structure that estimates the combined vertical load down from the roof material and snow, as well as their own weight. Further specifications can be found in the abridged *eifForm* XML model shown in Fig. 3.

Given an initial design from *Generative Components*, which in this example is the truss configuration shown in Fig. 5, an optimized design is generated in *eifForm* by using a planar truss structural grammar [9] to alter a combination of variables including the original structural member sizes, either allowing continuous values for size variables or selecting sections from a pre-defined set of sizes, point, or joint, locations that in turn modify the cantilever shape, and topology, which alters the structural configuration. Iterative modifications are made to the initial design within an optimization loop that seeks to generate optimally directed designs that are lightweight and adhere to modeled constraints (Fig. 1). In this example limits on maximum stress, member buckling, maximum relative displacement are defined and restrictions on the cantilever shape, modeled as geometric obstacles, are used. After design generation in *eifForm*, a set of optimized cantilevers are returned to *Generative Components* so that they can be interpreted within the context of the stadium as a whole (Fig. 6c and d).

For this example, considering the longest cantilever span with a length of 72.89 m, incrementally increasing the number of types of variables that can be modified, i.e. structural sizes, cantilever shape and cantilever topology, the structural mass is reduced. An initial optimization, for this span, allowing changes to section size alone, within a fixed set of available sections, yielded a mass of 48,283 kg while enabling a combined section size, shape and topology optimization led to a design with a mass of 35,036 kg (Fig. 7 middle). In the cases allowing shape and topology variation, geometric obstacles were used to influence the depth of the cantilever so that the designs produced do not obstruct visibility in the stadium. While reducing structural mass may not be a primary design goal, it is used here as a quantitative metric to compare design alternatives that all meet the complete set of modeled design criteria.

eifForm can also be used to explore relaxation of constraints in order to understand their impact on the designs produced. For example, relaxing the constraint on section sizes to allow continuous sizing of CHS sections with a fixed ratio of tube outer diameter to thickness equal to 14, rather than using only available CHS sections, resulted in generation of the set of optimally directed designs shown in Fig. 6c and d. The structural masses of these designs masses range from 17,816 kg for the shortest span (58.68 m) to 41,003 kg for the longest span (72.89 m). The design for the longest span, in this case, is also shown in Fig. 7 (top).

5. Discussion

A preliminary integration of two computational design systems, one performance-driven generative system and one associative geometry system, through

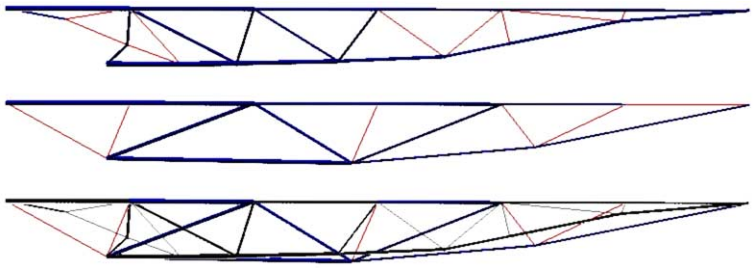

Fig. 7. Two cantilever truss alternatives for the same span that illustrate the design variation produced by relaxing design constraints: (top) novel, more tapered design with continuous CHS section sizes, 41,003 kg, (middle) more conventional, less tapered design that uses available CHS section sizes only, 35,036 kg (bottom) the two designs overlaid to illustrate the variation.

the use of XML models has been illustrated. Future extensions include:

- richer transfer of geometric data, parametric relations and constraints, as well as structural information between *Generative Components* and *eifForm*,
- building within *eifForm* on the customizable parametric geometric libraries in *Generative Components*,
- improved integration of XML formats and parsers between the systems, and
- use of *Generative Components* to gain more straight-forward definition, better understanding, and increased complexity of geometric modification rules that can then be transferred to *eifForm*.

The exploration of alternative, optimized stadium roof cantilever trusses was enhanced by being able to use *eifForm* within a larger parametric model of design context. It was found that due to the numerous constraints, mainly truss depth, most new designs produced did not vary drastically from the initial designs. In order to generate new designs (Fig. 5) that did not violate the design criterion of visibility, geometric obstacles were placed below the truss to drive the design generation towards a tapered solution. This shape does not occur naturally in the generation process as a deeper truss is most always lighter (Fig. 7 middle). Essentially the obstacles define the design envelope in rough way. A more robust and general way to enable the definition and control of design envelopes is underway and will allow points to be moved and added relative to a predefined boundary, which can be a line, curve, polygon, or surface. If the boundary changes then the related point, or joint, positions are updated. This technique has worked successfully in a specific implementation of STSA, the method within *eifForm*, for transmission tower redesign [12]. The numerous ways that points and curves can be created, related, and deleted in *Generative Components* can then be used as a means to define such geometric relations, gain an understanding of more complex shape modification rules and their parametric constraints and then transfer both the geometric model and new modification rule descriptions to *eifForm* so that subsequent generation processes reflect a desired style.

Associative geometry provides the capability for simple manipulation of the parametric model to dynamically consider alternative ‘what-if’ scenarios. Through improved and richer integration of XML models between the systems, both models can be created dynamically to take advantage of parametric capabilities. For example, major geometry that defines the stadium roof shape and dimensions can be altered, the new structural specification, including updated loading models, sent to *eifForm* that then generates a new set of optimized roof trusses in response to the modified geometry. This extended integration would provide dynamic feedback within the parametric model on structural impacts to changing geometry.

While only planar truss generation was shown in the example given, STSA is now capable of generating 3D roof truss systems, including truss-beams and interconnecting system bracing. Through the investigation presented here and further conversations with structural designers it was discovered that most would like a system that generates design alternatives which balance regularity and symmetry with spatial innovation. To achieve this effect without defining and

maintaining numerous types of symmetry constraints, a hierarchical composition model is used. Rather than generating truss-beams using tetrahedrons, it has been found that it is more effective to generate a planar design (truss A), copy and geometrically transform it to produce the other half of a truss-beam (truss B) and use a bracing algorithm to tie the two halves together (Fig. 8). Using the same planar truss grammar used to

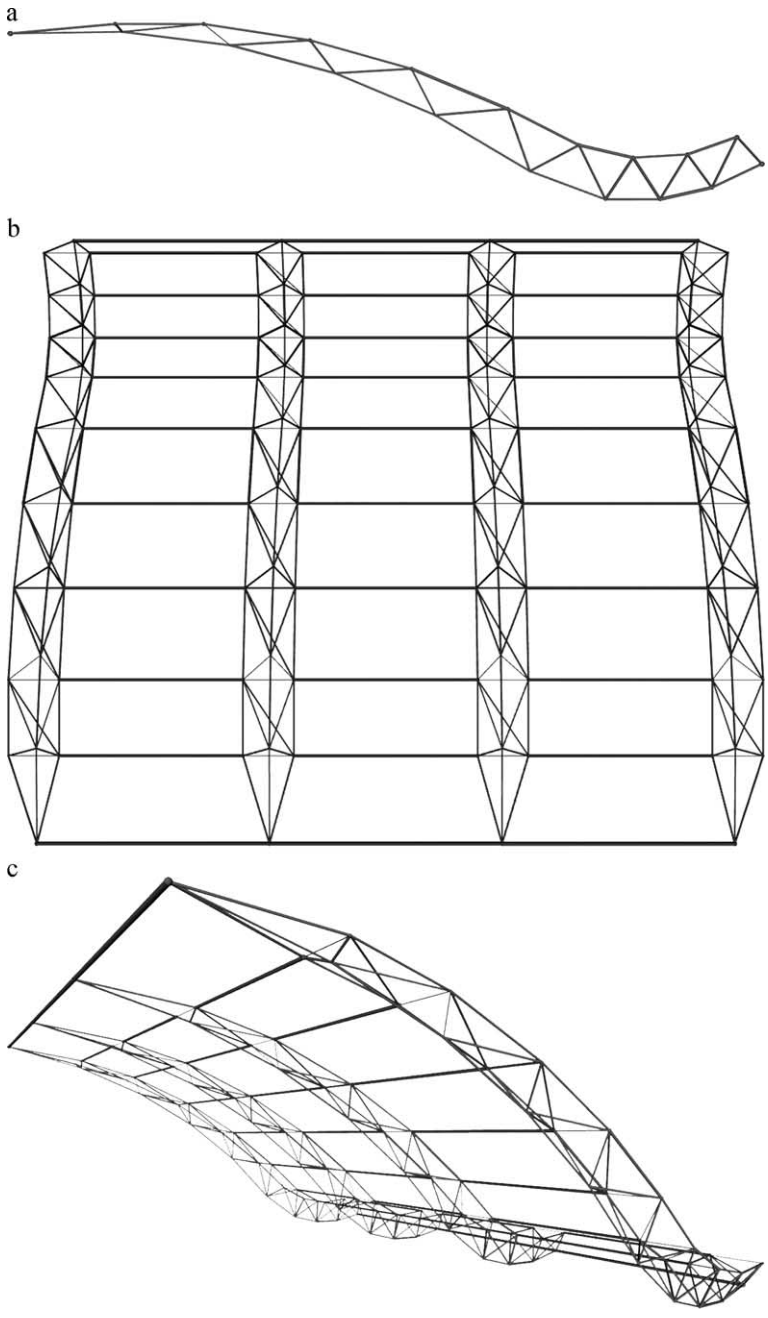

Fig. 8. A structural system of truss-beams generated using *eifForm* for a retrospective study of a Yanan schoolyard canopy in China.

create the designs shown in Figs. 6 and 7 [9] and enabling several possible geometric transformations of the design copy (truss B), e.g. translation, rotation, reflection, scale and their compositions, as well as using several bracing algorithms, a fairly comprehensive language of 3D truss structures can be produced from a very simple structural grammar.

This method extension has been used to design a light, truss-beam cantilever structure for a novel noon mark installation, which has recently been built [13]. A current retrospective study, which is relevant to the stadium roof truss example, is shown in Fig. 8 that depicts a cantilever truss-beam system generated for a schoolyard canopy design in China. In this case, reflecting the copied truss, truss B, about a vertical plane defined by the lower truss chord and applying a bracing algorithm generated individual truss-beams. Next, a simple system bracing algorithm was used to connect the individual truss-beams together thus generating a stable 3D roof truss system, which can be analyzed. Again, understanding of the geometric relations and modifications used in the generative method will be a key to its effective use and use for applications with increased complexity. Thus, further integration of *eifForm* and *Generative Components* will be pursued to take advantage of the available geometric libraries, associative modeling, and programming facilities for customization.

6. Conclusion

Integrated performance-driven generative design systems are aimed at creating new design processes that produce spatially novel yet efficient and buildable designs through exploitation of current computing and manufacturing capabilities. Integrating a generative structural design system, *eifForm*, within a standard CAD environment allows designers to experiment with the system within a familiar context and also allows creation of more complicated design scenarios for use as input models. Adding the extra capability of associative geometry, through the use of *Generative Components* in combination with *eifForm*, will enable designers to explore parametric variations of design scenarios dynamically and assess the structural impact of alternative building, enclosure and structural forms. The real challenge is to make systems that designers want to use in order to investigate the potential for performance-driven generative design to aid negotiations in multi-disciplinary design teams.

Acknowledgements

The authors would like to thank J. Parrish at Arup Sport for his collaboration on the stadium design example. The first and third authors would like to thank the EPSRC (UK), a Philip Leverhulme Prize awarded through the Leverhulme Trust (UK) and Bentley Systems for funding this research.

References

[1] R. Aish, Extensible computational design tools for exploratory architecture, in: B. Kolarevic (Ed.), Architecture in the Digital Age: Design and Manufacturing, Spon Press, London, 2003.
[2] R. Aish, Computer-aided design software to augment the creation of form, in: Francise Penz (Ed.), Computer in Architecture, Longman, 1992.
[3] L. Caldas, Evolving three-dimensional architecture form, in: J.S. Gero (Ed.), Artificial Intelligence in Design '02, Kluwer Academic Publishers, Dordrecht, 2002, pp. 351–370.
[4] B. Caporlette, Application integration using XML, Proceedings of XML Europe, Paris, France, 2000, http://www.gca.org/papers/xmleurope2000/index.html.
[5] J.P. Duarte, Customizing Mass Housing: A Discursive Grammar for Siza's Malagueira Houses, to appear in Automation in Construction, Special Issue on the eCAADe 03 Conference, Elsevier.
[6] J. Heisserman, Generative geometric design, IEEE Computer Graphics and Applications, 14, 1994, pp. 37–45.
[7] S. Kirkpatrick Jr, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–679.
[8] T. Knight, G. Stiny, Classical and non-classical computation, Architectural Research Quarterly 5 (2001) 355–372.
[9] K. Shea, J. Cagan, The design of novel roof trusses with shape annealing: assessing the ability of a computational method in aiding structural designers with varying design intent, Design Studies 20 (1999) 3–23.
[10] K. Shea, J. Cagan, Topology Design of Truss Structures by Shape Annealing, Proceedings of DETC98: 1998 ASME Design Engineering Technical Conference, September 13–16, 1998, Atlanta, GA19982001, DETC98/DAC-5624.
[11] K. Shea, J. Cagan, Innovative dome design: applying geodesic patterns with shape annealing. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 11 (1997) 379–394.
[12] K. Shea, I. Smith, Improving full-scale transmission tower design through topology and shape optimization, submitted to the ASCE Journal of Structural Engineering.

[13] K. Shea, X. Zhao, A novel noon mark cantilever support: from design generation to realization, To appear in the proceedings of IASS 2004: Shell and Spatial Structures from Models to Realization, 2004.
[14] G. Stiny, Introduction to shape and shape grammars, Environment and Planning. B 7 (1980) 343–351.
[15] W. Swartz, C. Sechen, New algorithms for the placement and routing of macro cells, Proceedings of the IEEE Conference on Computer-Aided Design, (Santa Clara, CA, November 11–15 1990), IEEE proceedings: Cat No. 90CH2924-9, 336–339.
[16] P. Szalapaj, CAD Principles for Architectural Design, Architectural Press, Oxford, 2001.