ON ENGINEERING DESIGN GENERATION WITH XML-BASED KNOWLEDGE-ENHANCED GRAMMARS

Stephan Rudolph
Similarity Mechanics Group, Institute for Statics and Dynamics of Aerospace Structures
University of Stuttgart, Germany
rudolph@isd.uni-stuttgart.de

Hansrudi Noser
Multimedia Laboratory, Department of Computer Science
University of Zürich, Switzerland
noser@ifi.unizh.ch

Abstract One of the bottlenecks in conceptual engineering design is the pure amount of design information which the design engineer needs to take into consideration. The design information is heterogeneous and consists of the design object behavior (i.e. the physics), its intended geometrical form and composition (i.e. the geometry) and miscellaneous other information pieces concerning manufacturing cost and more.
Despite the current transition in industry from former engineering drafting by hand to computer-aided design (CAD) software tools in 2D or 3D, even for evident aspects in conceptual design, i.e. the geometry and physics of the design object, the information processing chains of these two aspects of the early conceptual design effort is still not homogeneously supported by modern information technologies. One of the reasons for the missing links in the automation of design information processing is the lack of a unified design representation combining the geometrical and physical information about the design object.
This work explores some of the aspects of engineering design generation with XML-based knowledge-enhanced grammars and tries to identify the relative advantages and disadvantages of this automatic design generation technology.

Keywords: design generation, design representation, XML, L-systems, grammars

1. Motivation

Basic reason for the inadequate software support of the designer stems from the fact that the problem of conceptual engineering design is neither theoretically nor practically completely understood, nor exist appropriate complete representations and engineering information processing chains for conceptual design.

The problems associated with appropriate representations and suitable information processing chains for conceptual engineering design consist of the following main issues which occur simultaneously:

- *topologically* and *parametrically* different design solution concepts, for which envelopes or alternative enumeration schemes are not known, and
- *verbal, symbolic* and *numerical* pieces of information in different design models and at various degrees of detail.

Designers currently still communicate conceptual design information in many if not most cases by means of *human languages* which are able to deal with many aspects of the conceptual design problems mentioned above, the development of *XML-based knowledge-enhanced engineering design grammars* and the *corresponding virtual machine* is the main goal of this research.

The development of an *XML-based knowledge-enhanced engineering design grammar* involves many known aspects of grammar languages such as *syntax, semantic* and *pragmatic*. This opens the perspective for the reuse and adaptation of proven compiler techniques from computer science in the area of conceptual engineering design. The properties of the *XML-based knowledge-enhanced engineering design grammar* with a

- *syntax* allows the construction of expression parsers and design compilers as a function of the chosen design description, the
- *semantic* of the object model may incorporate geometrical as well as physical or any other information elements relevant in early conceptual design, and the
- *pragmatic* meaning of generic engineering design models in a specific conceptual design context allows for the efficient handling of a variety of different solution variants for the purpose of design evaluation.

2. Research Goals

The development of an *XML-based knowledge-enhanced engineering design grammar* will result in several strategic advantages over currently known conceptual design tools. These advantages are among others

- *unifying the representation* of physics and geometry in early conceptual design by integration of geometry and physics into the very same design object model,

- *bridging the gap* between different generic engineering design models with (but not limited to) object-oriented concepts, and
- *introducing the compilation* and automatic information processing chains into conceptual design for the purpose of design parameter selection, computation, design evaluation and visualization.

The development of an *XML-based knowledge-enhanced engineering design grammar* exhibits the following information flow shown in figure 1. The XML-description is parsed to an L-system description, of which the L-system parser separates the design object geometry and the design object equations for further separate processing.

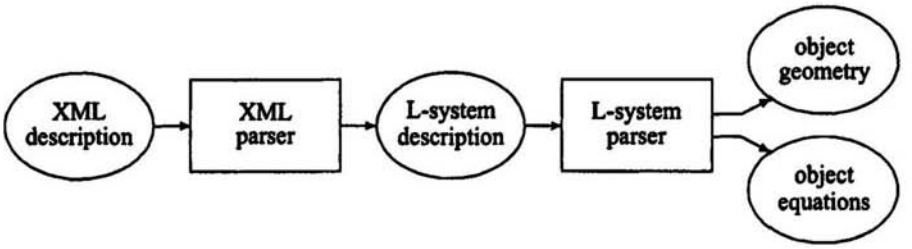

Figure 1. XML-based design grammar information processing chain

The benefit of the suggested approach lies in the editing of all engineering design knowledge in one single place using an *XML-based knowledge-enhanced engineering design grammar* instead of being tediously collected, adapted and assembled from different places and formats by the human designer. In the following, the information processing tasks of the parsers in figure 1 is described in more detail and the current state of the implementation is shown using some initial processing examples.

3. Preliminary Results

The subsequent parsers (L-system parser and the so-called solution path generator) already exist and work as visualized in the following. In respect to the upper branch of geometrical design information flow in figure 1, the design object geometry generated by the L-system grammar in figure 2 is shown. The e-system grammar displayed is hereby already the parsing result of an XML-based description grammar, which will be shown and discussed later following the overview in figure 6. More detailed descriptions of the turtle graphics used to visualize the resulting grammar string have been omitted here for reasons of space, but are well documented in (Prusinkiewicz & Lindenmayer 1996).

The result of the visualization process of the resulting grammar string is shown in figure 3. Various sources for the download of programs may be found on the Internet and in (Prusinkiewicz & Lindenmayer 1996).

$n = 7;\ \ w = 4;\ \ \delta = 22.5°;$
(global parsing parameters)

ω : A;
p_1 : A → [&FL!A] ///// [&FL!A] /////// [&FL!A] ;
p_2 : F → S ///// F ;
p_3 : S → F L ;
p_4 : L → [‘ ‘ ‘ ‘ ^^{-f+f+f-|-f+f+f}] ;

Figure 2. L-System *"Plant"* (Prusinkiewicz & Lindenmayer 1996)

Figure 3. L-System *"Plant"* (Prusinkiewicz & Lindenmayer 1996)

In respect to the lower branch of physical design information flow in figure 1, the design object equations are automatically generated and separated from the geometrical information.

3.1 Solution Path Generator

Design equations can be manipulated using graph-theoretic methods as shown in figure 4. As long as the design equations are "simple" enough, they can be handled and solved by a symbolic computer algebra program such as MAPLE V. Using the inbuilt code generation, C-code can be generated for the subsequent use in an efficient external optimization routine.

It is evident that the main advantage over other engineering approaches such as FEM (finite element method) software lies in the fact that the design equations may stem from *different* physical contexts which is typical for conceptual engineering design. In FEM programs the nature of the equations needs to be conform to the inbuilt element concept and the formal aspects of the avail-

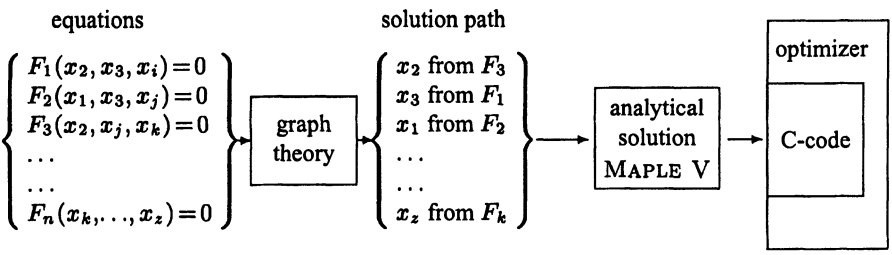

Figure 4. Solution path generator w. MAPLE V-back-end (Yusan & Rudolph 1999)

able matrix equation solver. Arbitrarily occurring and unstructured non-linear couplings are therefore difficult to deal with.

To illustrate the purpose of the solution path generator, the following (simple) input equations which describe the elementary thermodynamics of a gas turbine (Yusan & Rudolph 1999) are provided as input.

```
(T2/T1)-(P2/P1)^((kappa-1)/kappa)=0;      /* equations */
(T4/T3)-(P4/P3)^((kappa-1)/kappa)=0;
P2-P1*C=0;
P3-P2=0;
Wtp-mp*cp*(T4-T3)=0;
Qap+Qbp+Wnp=0;
P1-roh*R*T1=0;
Wnp-Wtp-Wcp=0;
Wcp-mp*cp*(T2-T1)=0;
Qbp-mp*cp*(T3-T2)=0;
P4-P1=0;
C := 4;          /* given design values and constants */
T1 := 273;
T3 := 1250;
mp := 1;
roh := 1.225;
kappa := 1.4;
R := 287;
cp := 1340;
```

The steps for the solution path generation are the following graph algorithms: *adjacency matrix representation, bipartite graph generation* (figure 5 a), *maximum bipartite graph matching* (figure 5 b), *directed graph of dependencies* (figure 5 c), *strongly connected component determination and isolation* (figure 5 d) and *reverse topological sorting* to obtain the solution path (Serrano 1987, Fertig & Reddy 1996).

Based on the *reversed topological sorting* of the last graph in figure 5 d, the solution path (i.e. the solution sequence) for the equations is determined. Using

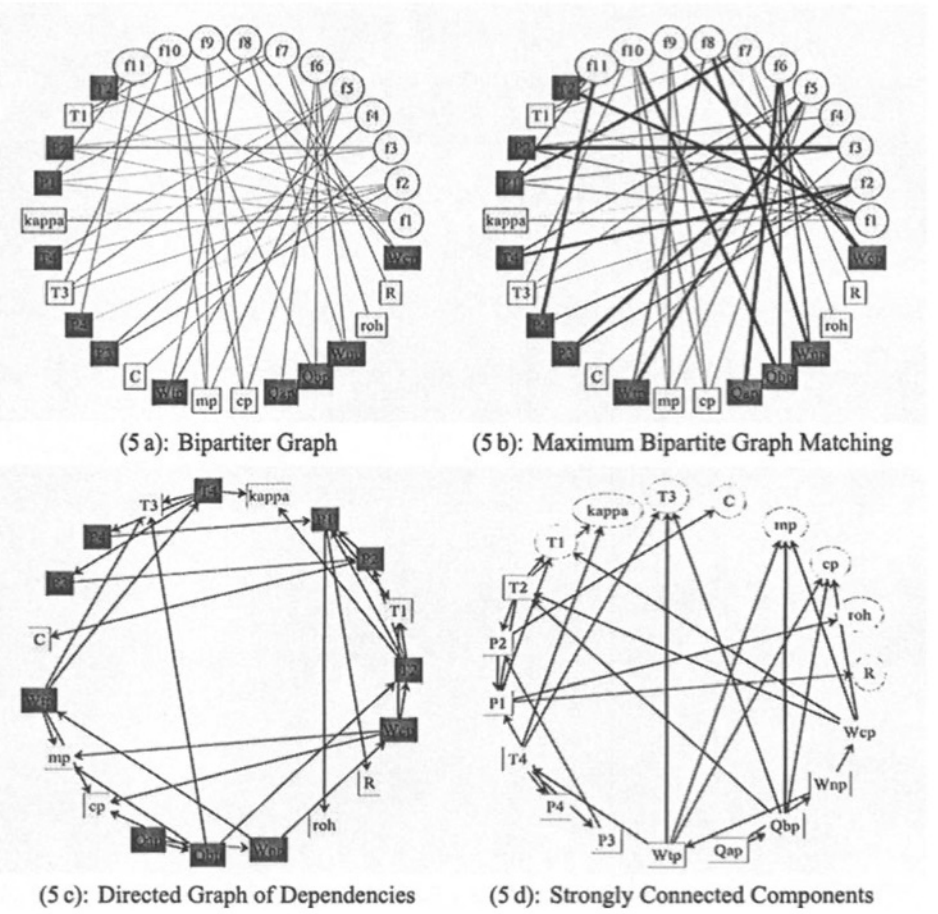

Figure 5. Sequence of graph algorithms

this information, the analytical solutions are computed through an interface to the computer algebra program MAPLEV:

```
f1 :=T2/T1-(P2/P1)^((kappa-1)/kappa) = 0   /* equations */
f2 :=T4/T3-(P4/P3)^((kappa-1)/kappa) = 0
f3 :=P2-P1*C = 0
f4 :=P3-P2 = 0
f5 :=Wtp-mp*cp*(T4-T3) = 0
f6 :=Qap+Qbp+Wnp = 0
f7 :=P1-roh*R*T1 = 0
f8 :=Wnp-Wtp-Wcp = 0
f9 :=Wcp-mp*cp*(T2-T1) = 0
f10 :=Qbp-mp*cp*(T3-T2) = 0
f11 :=P4-P1 = 0
P1 :=roh*R*T1                  /* solution path sequence */
```

```
P2 :=roh*R*T1*C
T2 :=C^((kappa-1)/kappa)*T1
P4 :=roh*R*T1
P3 :=roh*R*T1*C
T4 :=(1/C)^((kappa-1)/kappa)*T3
Wtp :=mp*cp*(1/C)^((kappa-1)/kappa)*T3-mp*cp*T3
Qbp :=mp*cp*T3-mp*cp*C^((kappa-1)/kappa)*T1
Wcp :=mp*cp*C^((kappa-1)/kappa)*T1-mp*cp*T1
Wnp :=-mp*cp*T1+mp*cp*(1/C)^((kappa-1)/kappa)*T3
      -mp*cp*T3+mp*cp*C^((kappa-1)/kappa)*T1
Qap :=mp*cp*T1-mp*cp*(1/C)^((kappa-1)/kappa)*T3
```

Using the previously indicated design values and constants, the following numerical values can also be determined using MAPLE V:

```
f1 :=T2/T1-(P2/P1)^((kappa-1)/kappa) = 0   /* equations */
f2 :=T4/T3-(P4/P3)^((kappa-1)/kappa) = 0
f3 :=P2-P1*C = 0
f4 :=P3-P2 = 0
f5 :=Wtp-mp*cp*(T4-T3) = 0
f6 :=Qap+Qbp+Wnp = 0
f7 :=P1-roh*R*T1 = 0
f8 :=Wnp-Wtp-Wcp = 0
f9 :=Wcp-mp*cp*(T2-T1) = 0
f10 :=Qbp-mp*cp*(T3-T2) = 0
f11 :=P4-P1 = 0
C :=4                          /* given design parameters */
T1 :=273
T3 :=1250
mp :=1
roh :=1.225
kappa :=1.4
R :=287
cp :=1340
P1 :=95979.97500                    /* numerical results */
P2 :=383919.9000
T2 :=405.6764409
P4 :=95979.97500
P3 :=383919.9000
T4 :=841.1876204
Wtp :=-547808.5887
Qbp :=1131393.569
Wcp :=177786.4308
Wnp :=-370022.1579
Qap :=-761371.4111
```

One disadvantage of this concept lies in the lower efficiency due to the versatility and more general symbolic capabilities of the back-end MAPLE V. Also, the reading of the XML-format of the knowledge-enhanced grammar seems to require some learning effort. It is however argued, that the advantage of concentrating all available design knowledge into one single file format compensates for this effort.

3.2 Current Implementation

The current status of the software implementation consists of the following software modules as indicated in figure 1: The `Java API` for `XML` parsing (available from `java.sun.com`) running under `JDK 1.2` or higher. The visualization of the generated object geometry is done using `Java3D API`.

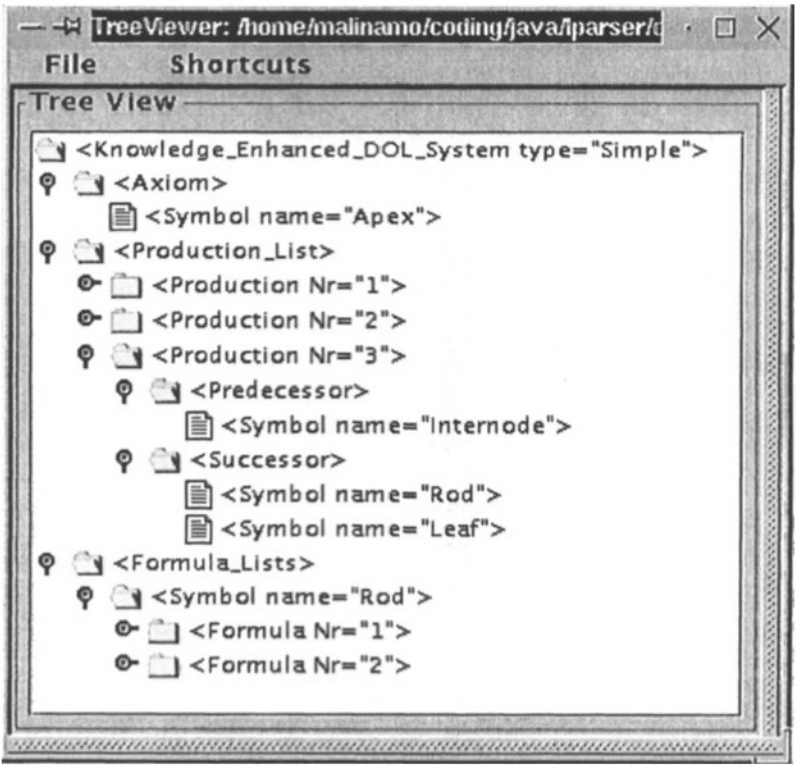

Figure 6. Structure of the XML-based object description

Figure 6 shows the principal structure of the `XML`-document description. According to the original L-system shown in figure 2, it consists of an axiom ω (here called `Apex`) and the production rules p_1 to p_3. The last production rule p_4 in figure 2 for the leaves is currently omitted in figure 6 for the reason of

simplicity. After the production rules for the geometry definitions, the formula list with two formulae is included for the description of the physics (force and momentum balance) of the design object.

The detailed XML-based object description is as following:

```
<?xml version="1.0"?>
<Knowledge_Enhanced_DOL_System type="Simple">
  <Axiom>
    <Symbol name="Apex" />
  </Axiom>
  <Production_List>
    <Production Nr="1">
      ....
    </Production>
    <Production Nr="2">
      <Predecessor>
        <Symbol name="Rod" />
      </Predecessor>
      <Successor>
        <Symbol name="Internode" />
        <Symbol name="/" />
        <Symbol name="/" />
        <Symbol name="/" />
        <Symbol name="/" />
        <Symbol name="/" />
        <Symbol name="Rod" />
      </Successor>
    </Production>
    <Production Nr="3">
      <Predecessor>
        <Symbol name="Internode" />
      </Predecessor>
      <Successor>
        <Symbol name="Rod" />
        <Symbol name="Leaf" />
      </Successor>
    </Production>
  </Production_List>
  <Formula_Lists>
    <Symbol name="Rod">
      <Formula Nr="1"> Force [
        <Parameter Name="symbolId" /> ] = mass_density *
          length * pow((thickness/2), 2) * Pi * vector [
        <Parameter Name="external_force" Component="x" />,
        <Parameter Name="external_force" Component="y" />,
        <Parameter Name="external_force" Component="z" />]
```

```
        + <VisitNeighbours Condition="Child"
        Operator="+">Force [ <Parameter Name="symbolId"
        Index="Neighbour" />] </VisitNeighbours>
      </Formula>
      <Formula Nr="2">
        ....
      </Formula>
    </Symbol>
  </Formula_Lists>
</Knowledge_Enhanced_DOL_System>
```

The generated L-system after parsing and some iterations looks like:

```
[ v Internode / / / / / Rod
Leaf !  [ v Rod
Leaf !  Apex ] / / / / / c [ v Rod
Leaf !  Apex ] / / / / / c [ v Rod
Leaf !  Apex ] ] / / / / / c [ v Internode / / / / / Rod
Leaf !  [ v Rod
Leaf !  Apex ] / / / / / c [ v Rod
Leaf !  Apex ] / / / / / c [ v Rod
Leaf !  Apex ] ] / / / / / c [ v Internode / / / / / Rod
Leaf !  [ v Rod
Leaf !  Apex ] / / / / / c [ v Rod
Leaf !  Apex ] / / / / / c [ v Rod
Leaf !  Apex ] ]
```

Parsing the list of physical formulae yields the following expressions:

```
 Force[0]=mass_density*1.7*(0.2/2)*(0.2/2)*Pi*vector([0.0,-1.0,0.0])+
          Force[1]+Force[2]+Force[3]
Moment[0]=crossprod(scalarmul(vector([0.0,0.95105654,-0.309017]),
          (1/2)*1.7),scalarmul(vector([0.0,-1.0,0.0]),mass_density*1.7
          *(0.2/2)*(0.2/2)*Pi))+crossprod(scalarmul(vector([0.0,
          0.95105654,-0.309017]),1.7),Force[1])+crossprod(scalarmul(
          vector([0.0,0.95105654,-0.309017]),1.7),Force[2])+
          crossprod(scalarmul(vector([0.0,0.95105654,-0.309017]),
          1.7),Force[3])+Moment[1]+Moment[2]+Moment[3]
          ....
```

Remark: At the time of writing, the focus of the above equations (`Force[0]` = $\rho_g \cdot l \cdot (d/2)^2 \cdot \pi \cdot$ `v[0]` + *external forces*) lies in their automatic generation. This expresses that the forces of the static equilibrium depend on the weight (*mass_density times length times cross section*) of each rod plus the external forces from other "child" rods. Furthermore, closing the gap in the present design equation example is obvious if the similarity of the above equations is compared to the equation input file of the gas turbine example in section 3.1.

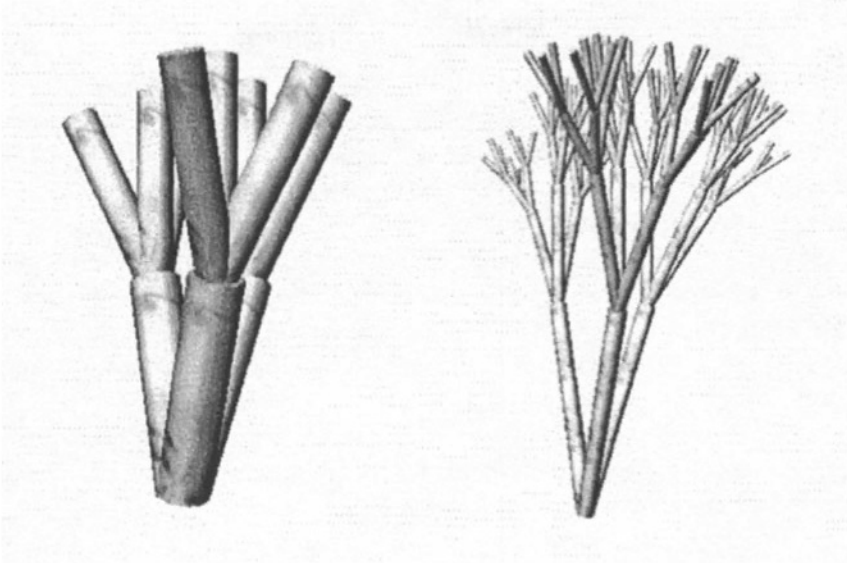

Figure 7. Expansion of tree-like L-system with $n=2$ (left), $n=4$ (right)

Based on the previous XML-parsing result, the geometry of the design object can be generated by the L-system parser. Figure 7 shows the iteration depths $n=2$ and $n=4$ of the generated L-system grammar.

3.3 Discussion

XML-based design object descriptions offer, independent of their underlying philosophical conception as biologically inspired *"L-system"* or more conventional mechanical *"design feature"*, the advantage of a program and product independent format definition. With the increasing relevance of computer networks in the engineering design and development chain this will become more and more important.

Problems associated with this approach are however manifold: Format independence reduces the compactness and efficiency with which the universal knowledge representation will be processed. In addition, as the representation length grows, readability is reduced and the complexity of the different assembled and unified model representations increases as well. In the current implementation the design equations are restricted to closed algebraic forms, in theory differential and integral equations are also permissible. The term *equation* is used here as a substitute for the term *constraint* as is typically used in the area of *constraint processing* (Serrano 1987).

Furthermore, geometry appears more difficult to handle properly than complex equations and boundary conditions. On inspection of figure 7, it is evident

that the intersection of the rods requires still more effort to obtain a design object which can be manufactured. Since L-systems are mainly topological geometry descriptions, more (production) rules and symbols are needed to create acceptable and consistent geometries for mechanical design purposes.

The main argument of a knowledge-enhanced grammar for engineering design has an analogy in computer science: Donald Knuth, the creator of the TEX-text-formatting language emphasized this argument in software engineering and created WEB, a meta language with can be parsed to yield separate "C" and "TEX" data files using the `cweave` and `ctangle` programs. Knuth argued that only the integration of the program documentation into the program source code file would keep both program and documentation consistent (see the processing analogy in figure 1) with each other. Knuth called this programming paradigm "literate programming".

It is felt by the authors that because of the complexity of the task of engineering design an analogous argument holds for the purpose of engineering design generation and its related documentation.

4. Summary

An XML-based knowledge-enhanced grammar is suggested to create an unified design object representation which combines geometrical and physical information. The relative advantages and disadvantages of the resulting automatic design generation and information processing chain are discussed. XML-based design representations foster the exchange of data and information across programs and platforms.

The approach may also nicely complement currently available CAD software tools through appropriate *interfaces* to these CAD-systems or Finite-Element-Method (FEM) Analysis software tools, since the automatic geometry generation could substitute the manual generation of a geometric description which is still a tedious and error-prone task.

Acknowledgements

The authors thank the students Michael Bölling, Hakan Yusan and Marijo Malinar for their programming support in this research.

References

Fertig, K. W. and Reddy, Y. S. (1996) *Constraint Management Methodology for Conceptual Design Tradeoff Studies.* Proceedings 1996 ASME Design Engineering Technical Conference, August 18-22, Irvine, CA, USA.

Prusinkiewicz, P. and Lindenmayer, A. (1996) *The Algorithmic Beauty of Plants*, Springer Verlag, Berlin.

Serrano, D. P. (1987) *Constraint Management in Conceptual Design*, PhD Thesis, Department of Mechanical Engineering, MIT, Cambridge, MA, USA.

Yusan, H. and Rudolph, S. (1999) *A Study of Constraint Management Integration into the Conceptual Design Phase*, Proceedings 25th Design Automation Conference, Las Vegas, NV, September 12-15.