10TH INTERNATIONAL DESIGN STRUCTURE MATRIX CONFERENCE, DSM'08
11 – 12 NOVEMBER 2008, STOCKHOLM, SWEDEN

SYNTHESIS OF PRODUCT ARCHITECTURES USING A DSM/DMM-BASED APPROACH

David Wyatt, David Wynn and John Clarkson
Cambridge Engineering Design Centre, University of Cambridge

Keywords: Product architecture, Component DSM, constraint, computational synthesis

1 INTRODUCTION

The Component Design Structure Matrix (DSM) is an established tool for studying product architectures. Previous research using Component DSMs has mainly focused on capturing and analysing existing product architectures (e.g. [1]). Methods which use DSMs to support the design of better product architectures are less common. This short paper proposes that the design of better products can be supported by a method which captures the rules governing architectures that "make sense", and uses these rules to compare a particular architecture concept represented in a Component DSM against the space of possible architectures for that product. We describe a constraint model that uses DSMs and DMMs to specify these rules and an algorithm for generating and evaluating alternative Component DSMs which satisfy the constraints. The architecture synthesis approach has been implemented in a prototype tool, which is illustrated through application to a simple problem.
The motivation for the research is elaborated in Section 2. The computational approach for architecture design is outlined in Section 3. Section 4 illustrates the method using an example, and Section 5 draws conclusions and indicates further work.

2 MOTIVATION

Decisions about product architecture have great significance for the cost, quality and performance of an engineering product. It is therefore critical that appropriate decisions are made when designing product architectures, which must be done early in the design process. However, architecture decisions are some of the most difficult to make in design, due to the large number of possible architectures, the complex constraints governing their feasibility, as well as the high levels of uncertainty often present in the early stages of design. In this paper, we argue that formal computer-based methods for architecture design have the potential to support these difficult decisions by generating a wide variety of possible architectures and helping the designer evaluate these alternatives objectively.
As well as assisting with the design of original product architectures, we propose that such methods could support design-by-redesign, which is common practice for complex products [2]. In particular, computational synthesis may be used to investigate the scope for improvements to an existing architecture by comparing it with the alternatives that can be obtained through incremental change. In such cases, either the regions of the architecture to be redesigned could be indicated by the designer or the existing architecture could be iteratively "mutated" by computer to explore alternatives.

3 APPROACH

"Architecture" has been used to refer to many aspects of product structure. In this paper we consider the pattern of *connections* between the *components* of a product (the *topology* of the product), represented as a Component DSM. This definition corresponds to the third of the three elements of product architecture defined by Ulrich [3] (the other two are a) the way an overall function is decomposed into an arrangement of functional elements, and b) the way functional elements are embodied by components). We focus on topology, the 'least fundamental' element of product architecture, since it requires the least investment to vary. In our model, components have *component types* to indicate interchangeability (e.g., engines on aircraft), and connections have *connection types*.
There are many potential topologies for an artefact described in this way; however, the majority of these are infeasible in that they do not "make sense" and could not serve as the architecture of a real product – they are not "feasible". Our method defines *feasibility* through two classes of constraints:

1. *Connection requirements* ('syntax') specify which connections a given type of component may have, in terms of minimum and maximum degrees for each type of connection. For example, a

component might require between 1 and ∞ "attached to" connections, indicating that it *must* be attached to at least one other component but *may* be attached to an unlimited number.

2. *Path requirements* ('semantics') specify paths that must exist in the Component DSM. These constrain 'long-range' properties of the DSM's connections, unlike the 'local' connection requirements. Path requirements are defined by the component types between which a path must exist and the connection types that may constitute the path, and are related to the overall function of the product. For example, in a hairdryer with overall function "produce a flow of hot air", there must be an airflow path from the heater to the outlet nozzle.

The ontology of components, component types and connection types, combined with the constraints, define an *architecture schema.*

To generate candidate architectures from such a schema, an exhaustive breadth-first search is carried out. The search starts from an empty Component DSM, and connections are added one-at-a-time (up to a specified maximum search depth). Once all connection requirements (minimum and maximum) are satisfied, graph-search algorithms are used to test the path requirements. The resulting list of feasible architectures may then be reviewed by the designer to check that the schema is suitable, i.e., that it results in possibilities that "make sense". Each feasible architecture is then evaluated against objectives. In this paper, we consider two objectives: 'changeability', i.e. immunity from change propagation, calculated as the average of the shortest graph distances between all pairs of components (based on [4]); and 'designability', or design effort required, calculated as the skewness of the distribution of numbers of interfaces per component. These objectives are used to present the feasible architectures to the designer on a Pareto plot to allow a final selection to be made. This method may also be used for design based on an existing architecture by starting the search with a partially filled Component DSM representing the parts of the architecture to be conserved; the computational search generates options for the remainder.

4 ILLUSTRATIVE EXAMPLE

Figure 1 shows the approach applied to aeroplane configuration. The top row of shows the architecture schema, which consists of: a DMM to list components and relate them to component types; a DMM between component types and connection types, specifying the numerical connection requirements; and a DSM of component types, whose entries indicate the path requirements (both where paths are required and the allowable connection types). The schema and the search algorithm were implemented using the Scheme programming language. The middle row of the figure shows the architectures of three commercial aircraft that were independently re-invented by the search process. The lower part of the figure shows the 'designability' and 'changeability' (as defined above) of all 375 feasible architectures resulting from this schema. Many of these alternatives coincide on the Pareto plot due to the poor resolution of the evaluation methods used.

5 CONCLUSIONS

In many domains, a design's overall performance is strongly influenced by its architecture. This paper has proposed that computational synthesis of alternative architectures could help designers produce 'better' designs. A constraint model was introduced that defines the 'feasibility' of a product architecture represented as a Component DSM. An algorithm for synthesising architectures conforming to these constraints was described, and a simple example of aircraft configuration was presented to demonstrate the ideas. The main contribution of this paper is to show one way in which DSM tools can directly support the creation of better designs – as well analysing existing designs.

In addition to designing new product architectures, we have argued that the same computational synthesis approach could potentially be used to identify beneficial improvements to existing product architectures. It could also support the design of systems in other domains, such as organisations and processes, whose architectures can be described in a similar way to those of products.

To tackle more realistic problems a number of issues require further work. These include:

- <u>More detailed representations of architecture:</u> The information contained in a binary Component DSM is not sufficient to assess many of the objectives relevant to design [5]. Additional domains, entity types and attributes may need to be added to the information in the Component DSM.
- <u>Implementation:</u> We envisage a generally-applicable tool for architecture synthesis and evaluation, allowing intuitive specification of constraints and evaluation methods for different types of system. Research is ongoing to explore how this can be achieved.

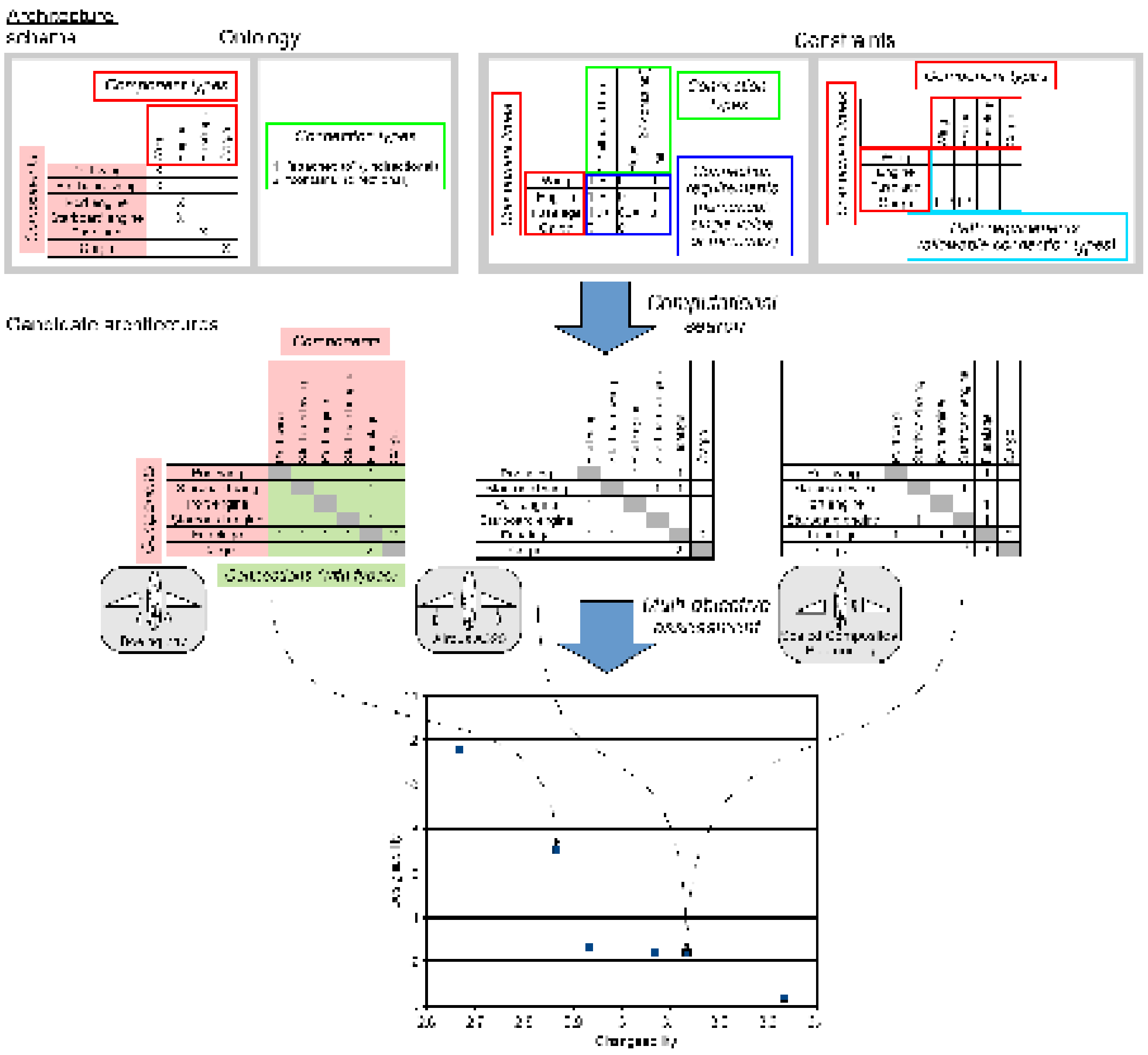


Figure 1. The aircraft architecture schema and the results of the computational search process.

REFERENCES

[1] Sosa M.E., Agrawal A., Eppinger S.D. and Rowles C.M. A network approach to define modularity of product components. In *Proceedings of the ASME Design Engineering Technical Conference and Computers and Information in Engineering Conference (IDETC/CIE 2005)*, Long Beach, California, USA, September 24-28 2005.

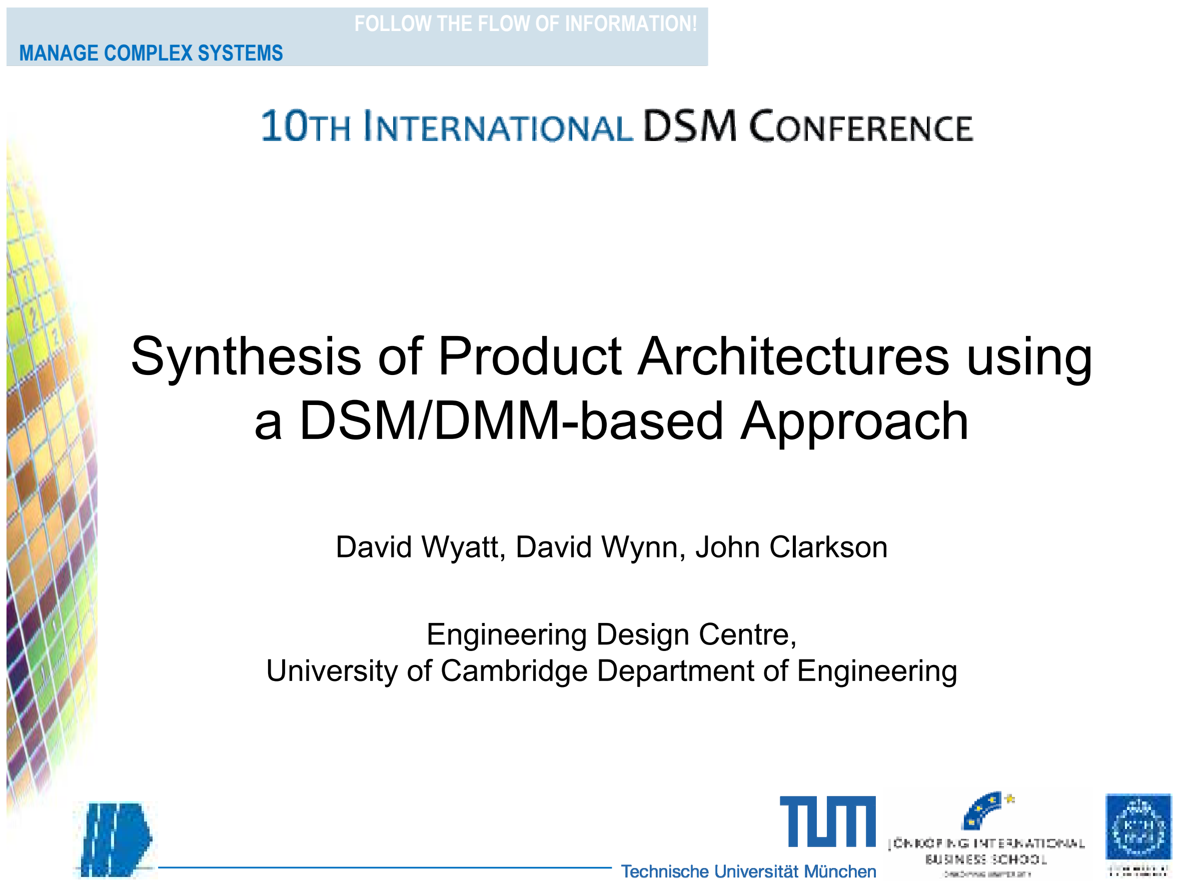
[2] Eckert C.M., Clarkson P.J. and Zanker W. Change and customisation in complex engineering domains. *Research in Engineering Design*, 2004, 15(1), pp. 1-21.

[3] Ulrich K. The role of product architecture in the manufacturing firm. *Research Policy*, 1995, 24(3), pp. 419-440.

[4] Keller R., Eckert C.M. and Clarkson P.J. Heuristics for Change Prediction. In *Proceedings of the International Design Conference (DESIGN 2006)*, Dubrovnik, Croatia, 2006, pp. 873-880.

[5] Wyatt D.F., Wynn D.C. and Clarkson P.J. Comparing representations for system architecture design through life-cycle evaluation methods. In *Proceedings of the 2nd Nordic Conference on Product Lifecycle Management*, Göteborg, Sweden, January 28-29 2009 (to appear).

Contact: David F. Wyatt
Cambridge Engineering Design Centre
Department of Engineering, University of Cambridge
Trumpington Street, Cambridge CB2 1PZ, UK
+44 (0)1223 332 673
dw274@cam.ac.uk


MANAGE COMPLEX SYSTEMS

FOLLOW THE FLOW OF INFORMATION!

10TH INTERNATIONAL DSM CONFERENCE

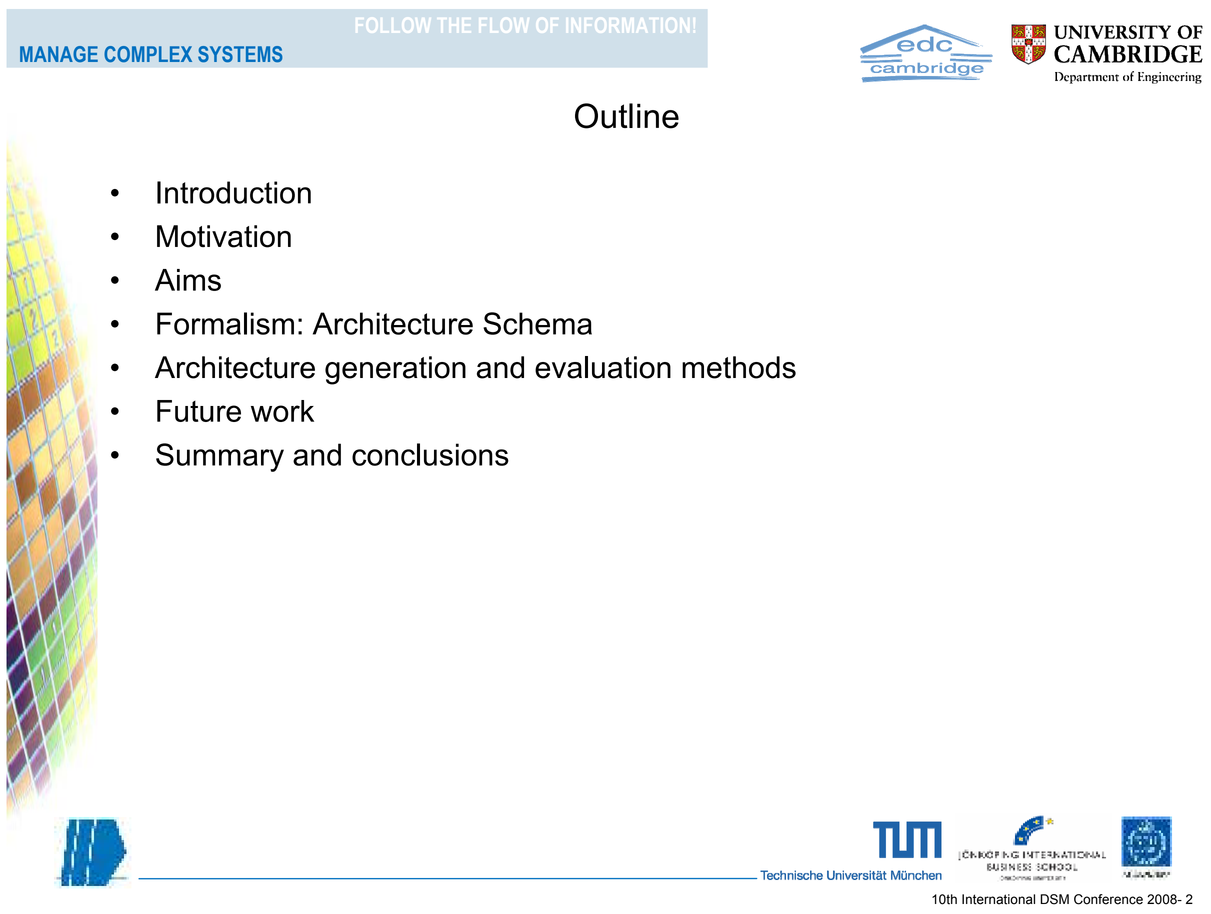
Synthesis of Product Architectures using a DSM/DMM-based Approach

David Wyatt, David Wynn, John Clarkson

Engineering Design Centre,
University of Cambridge Department of Engineering

Technische Universität München

JÖNKÖPING INTERNATIONAL BUSINESS SCHOOL


MANAGE COMPLEX SYSTEMS

FOLLOW THE FLOW OF INFORMATION!

edc cambridge

UNIVERSITY OF CAMBRIDGE
Department of Engineering

Outline

- Introduction
- Motivation
- Aims
- Formalism: Architecture Schema
- Architecture generation and evaluation methods
- Future work
- Summary and conclusions

Technische Universität München

JÖNKÖPING INTERNATIONAL BUSINESS SCHOOL

Introduction

- Research to understand product architecture design
  - Levels of architecture
    - Functions
    - Modules
    - Components
    - Key parameters
    - …
  - Phases in design
    - Generate
    - Evaluate
    - Select

Product architecture

- Karl Ulrich, 1995, "The role of product architecture in the manufacturing firm", *Research Policy* 24(3) pp. 419-440:
  1. the arrangement of *functional elements;*
  2. the mapping from *functional elements* to *physical components;*
  3. the specification of the *interfaces* among interacting physical components.
  - Represent as DSM

Motivation

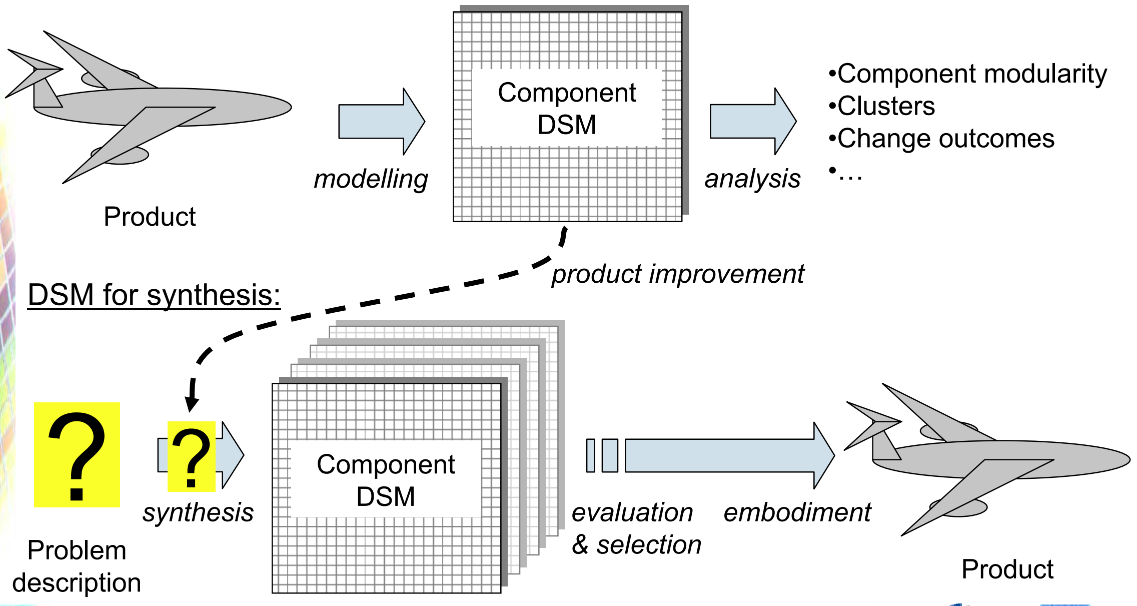
- Important but difficult to design architectures
  - Up to 80% of lifecycle cost committed
  - Combinatorial possibilities with complex constraints
  - Pre-existing product (platform) issues

Aims

- Provide a formal set of constraints to determine whether a given component DSM is “feasible”
- Implement computationally for sample problems

Aims (2)

DSM for analysis:

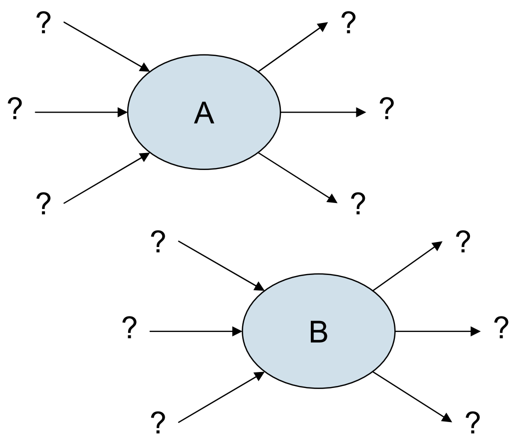


DSM for synthesis:

Formalism: Architecture schema

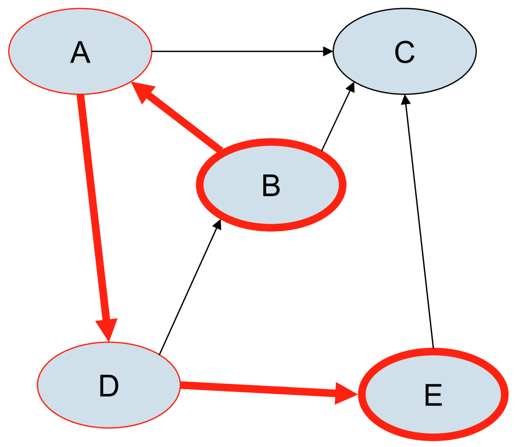
- Ontology
  - Components and types
  - Connection types
- Constraints
  - Connection requirements
  - Path requirements

Connection requirements



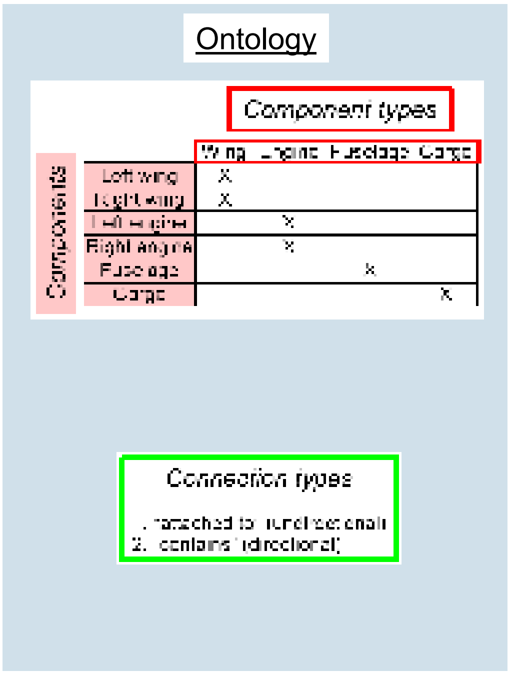
- Minimum and maximum degrees for each connection type for a given type of component
- “What can be connected to what, and how many times”
  - “Syntax”
- Example (hairdryer):
  - power supply:
    - “attached to” = 1
    - “supplies electricity to” = 0…∞
- Represent as DMM

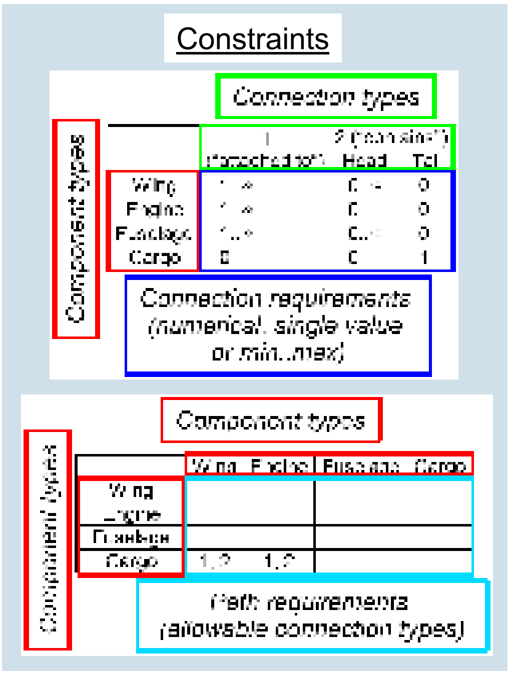
Path requirements



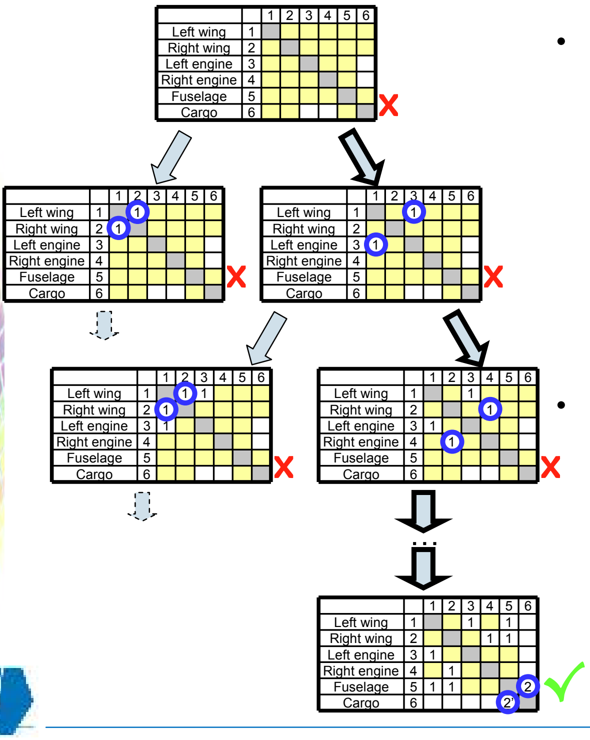
- Paths (of particular connection types) that must exist in the component DSM
- Related to overall function
- “What has to be linked to what, and how”
  - “Semantics”
- Example (hairdryer):
  - Function “produce hot air” ⇒ path of connection-type “airflow” from “heater” to “nozzle”
- Represent as DSM

Architecture schema example: aeroplanes





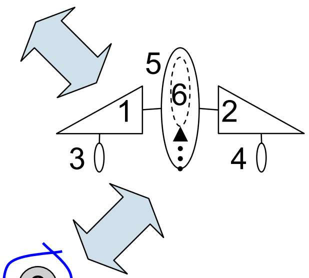
Architecture generation algorithm

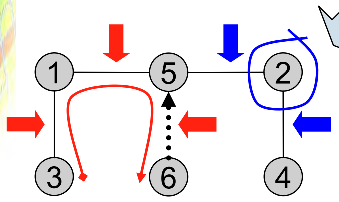


- Breadth-first search:
  - Start with blank DSM
  - Add connections one-at-a-time
  - Once connection requirements fulfilled, check path requirements
  - Search exhaustively up to a maximum depth
- Implementation
  - Written in Scheme

Example architecture evaluation methods

		1	2	3	4	5	6
Left wing	1	▨		1		1	
Right wing	2		▨		1	1	
Left engine	3	1		▨			
Right engine	4		1		▨		
Fuselage	5	1	1			▨	2
Cargo	6					2'	▨





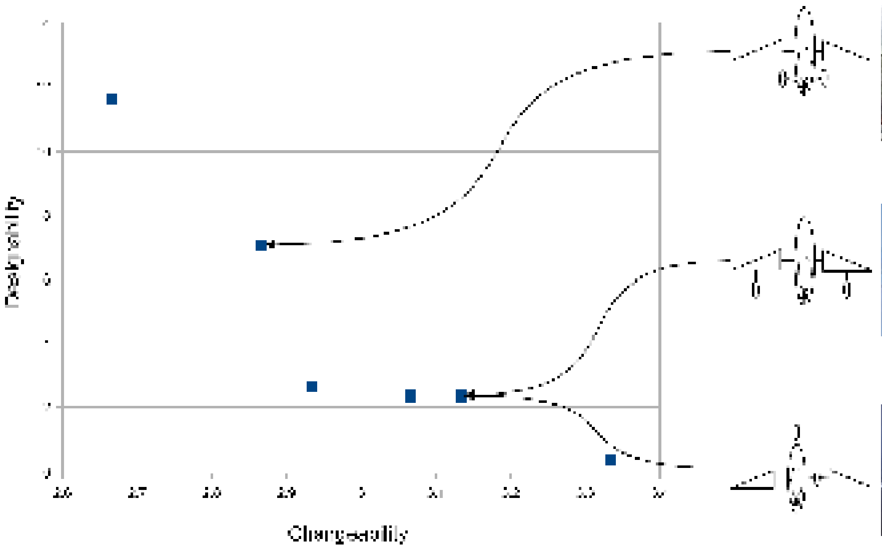
Number of interfaces of component 2, $z_2 = 2$

Shortest graph distance from 3 to 6, $d_{36} = 3$

- Immunity from change propagation ("Changeability"):
  - Average graph distance between pairs of components
    - René Keller, Claudia Eckert & John Clarkson, 2006, "Heuristics for Change Prediction", in *Proceedings of the International Design Conference (DESIGN 2006)*
- Design effort requirement ("Designability"):
  - Skewness of distribution of number of interfaces per component

Results: aeroplanes

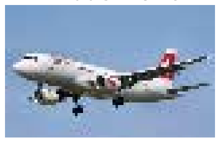
- 375 possible architectures
- 15 seconds on 2.2GHz Athlon with 1Gb RAM



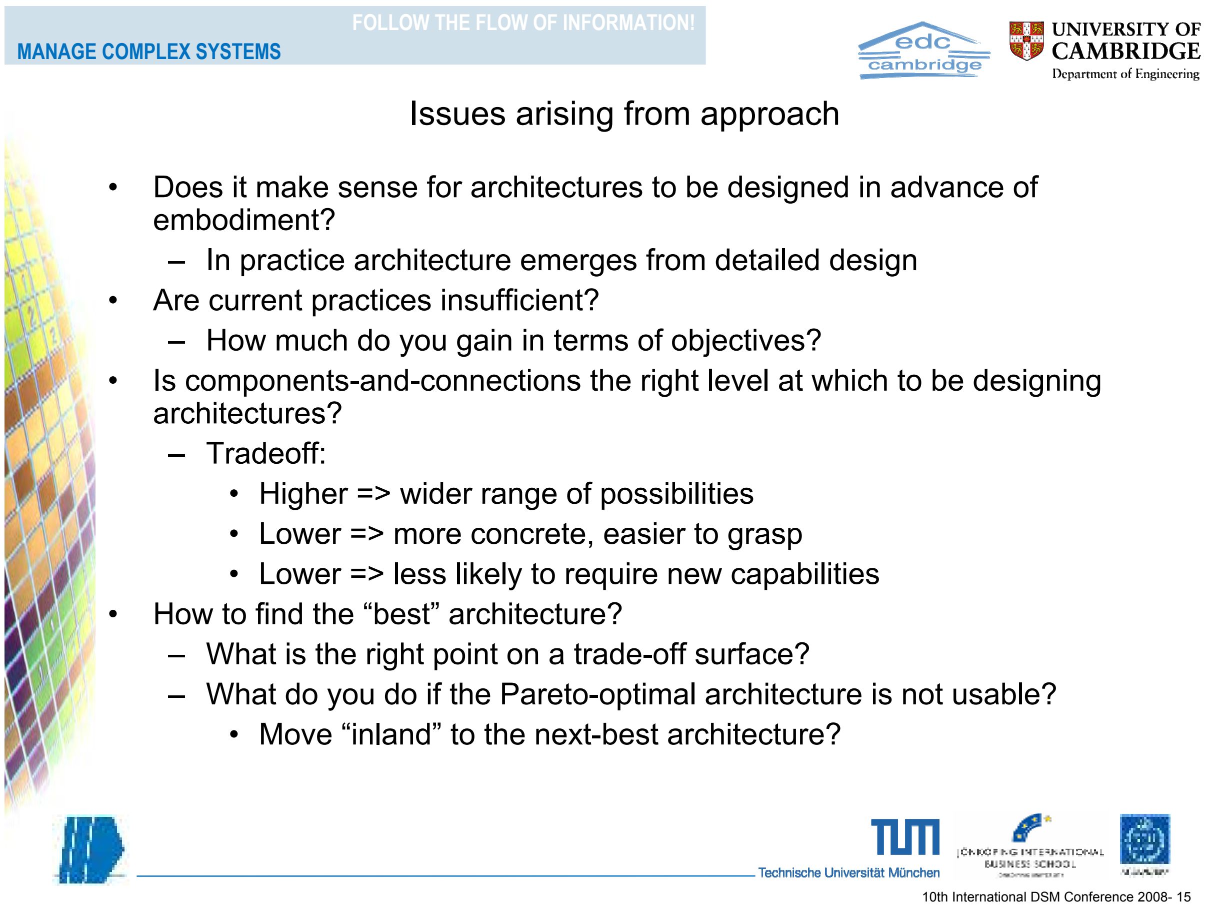
Corresponding real aircraft:

Boeing 717

Airbus A320



SC Boomerang



MANAGE COMPLEX SYSTEMS

FOLLOW THE FLOW OF INFORMATION!

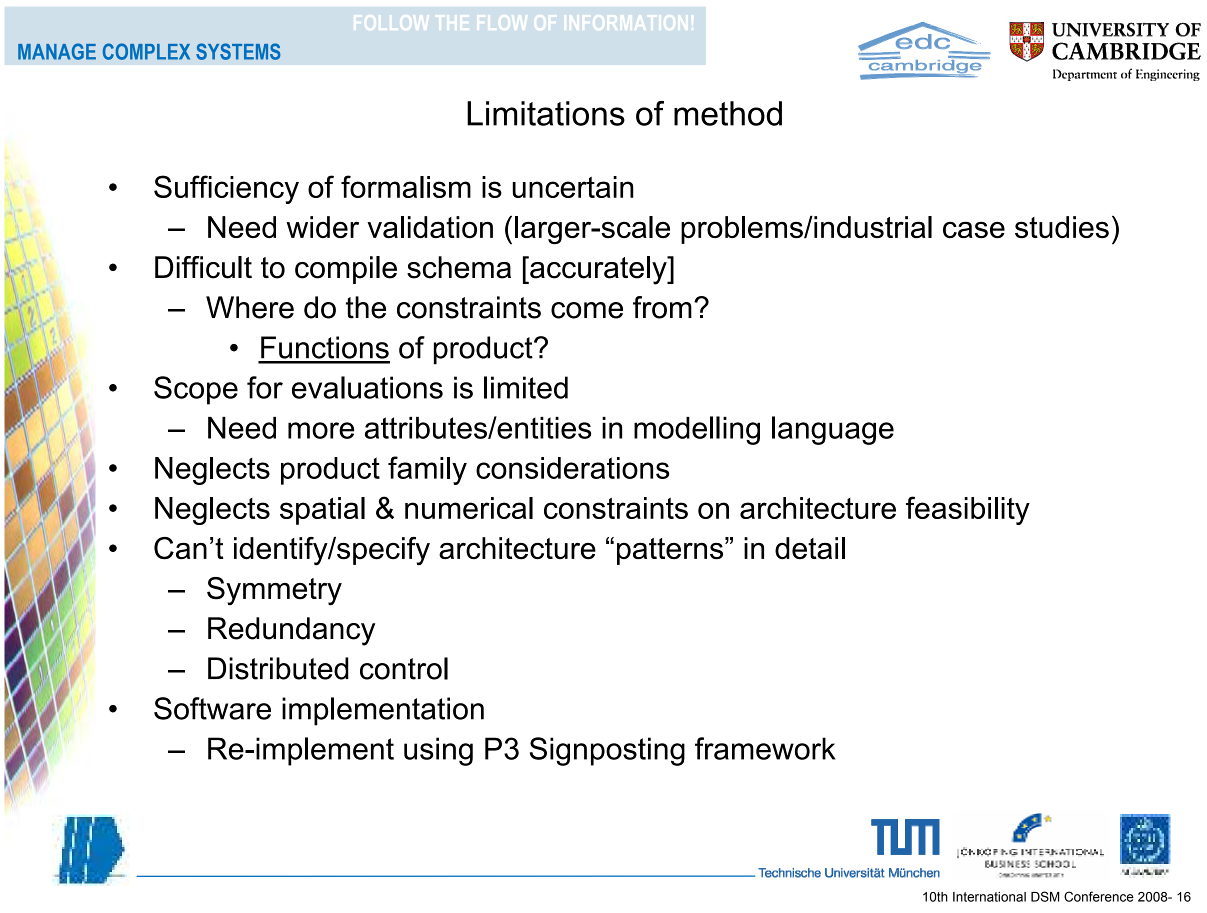
UNIVERSITY OF CAMBRIDGE
Department of Engineering

Issues arising from approach

- Does it make sense for architectures to be designed in advance of embodiment?
  – In practice architecture emerges from detailed design
- Are current practices insufficient?
  – How much do you gain in terms of objectives?
- Is components-and-connections the right level at which to be designing architectures?
  – Tradeoff:
    - Higher => wider range of possibilities
    - Lower => more concrete, easier to grasp
    - Lower => less likely to require new capabilities
- How to find the “best” architecture?
  – What is the right point on a trade-off surface?
  – What do you do if the Pareto-optimal architecture is not usable?
    - Move “inland” to the next-best architecture?

Technische Universität München

10th International DSM Conference 2008- 15



MANAGE COMPLEX SYSTEMS

FOLLOW THE FLOW OF INFORMATION!

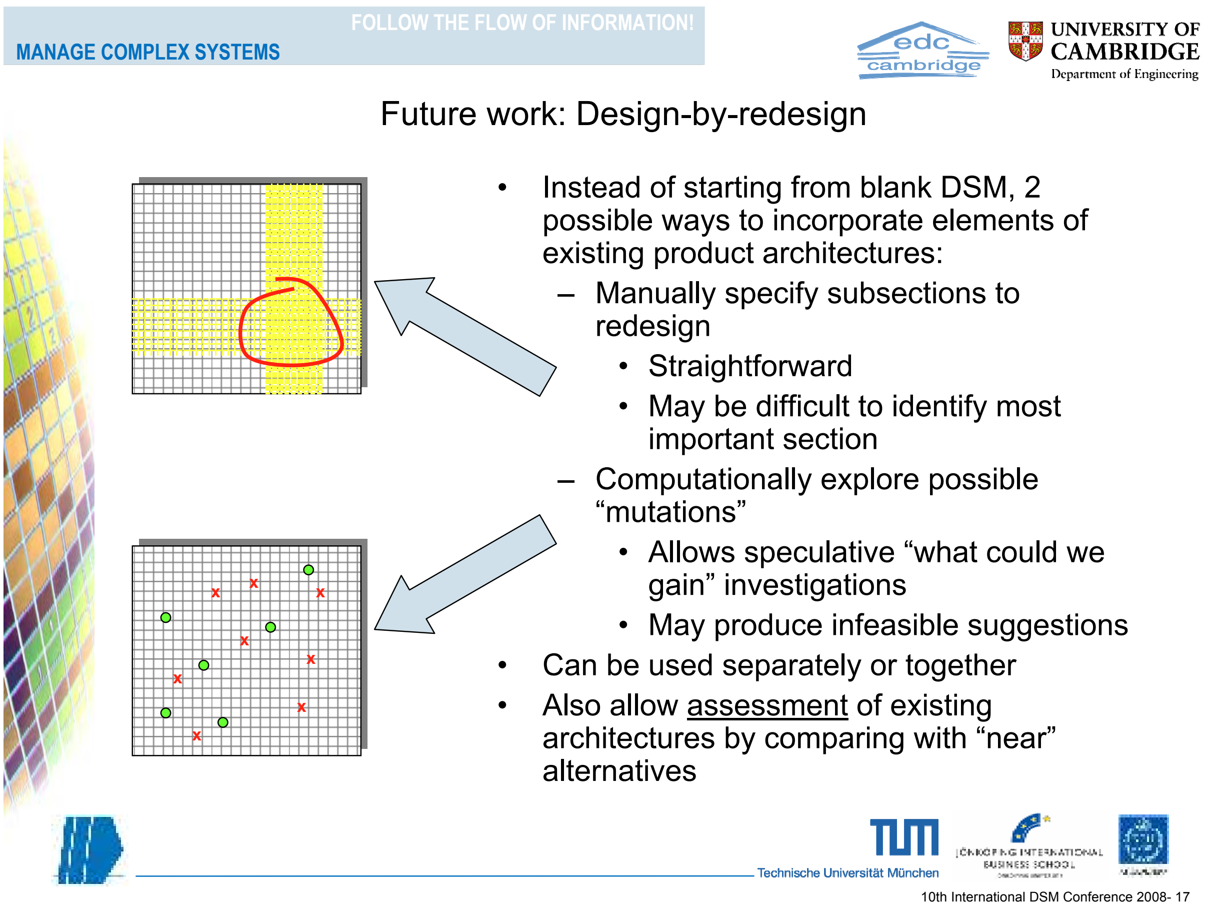
UNIVERSITY OF CAMBRIDGE
Department of Engineering

Limitations of method

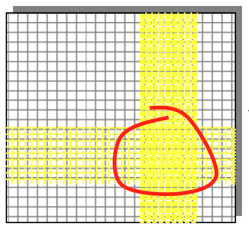
- Sufficiency of formalism is uncertain
  – Need wider validation (larger-scale problems/industrial case studies)
- Difficult to compile schema [accurately]
  – Where do the constraints come from?
    - Functions of product?
- Scope for evaluations is limited
  – Need more attributes/entities in modelling language
- Neglects product family considerations
- Neglects spatial & numerical constraints on architecture feasibility
- Can’t identify/specify architecture “patterns” in detail
  – Symmetry
  – Redundancy
  – Distributed control
- Software implementation
  – Re-implement using P3 Signposting framework

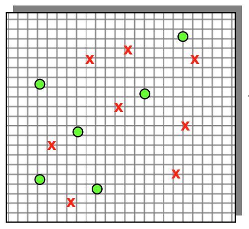
Technische Universität München

10th International DSM Conference 2008- 16

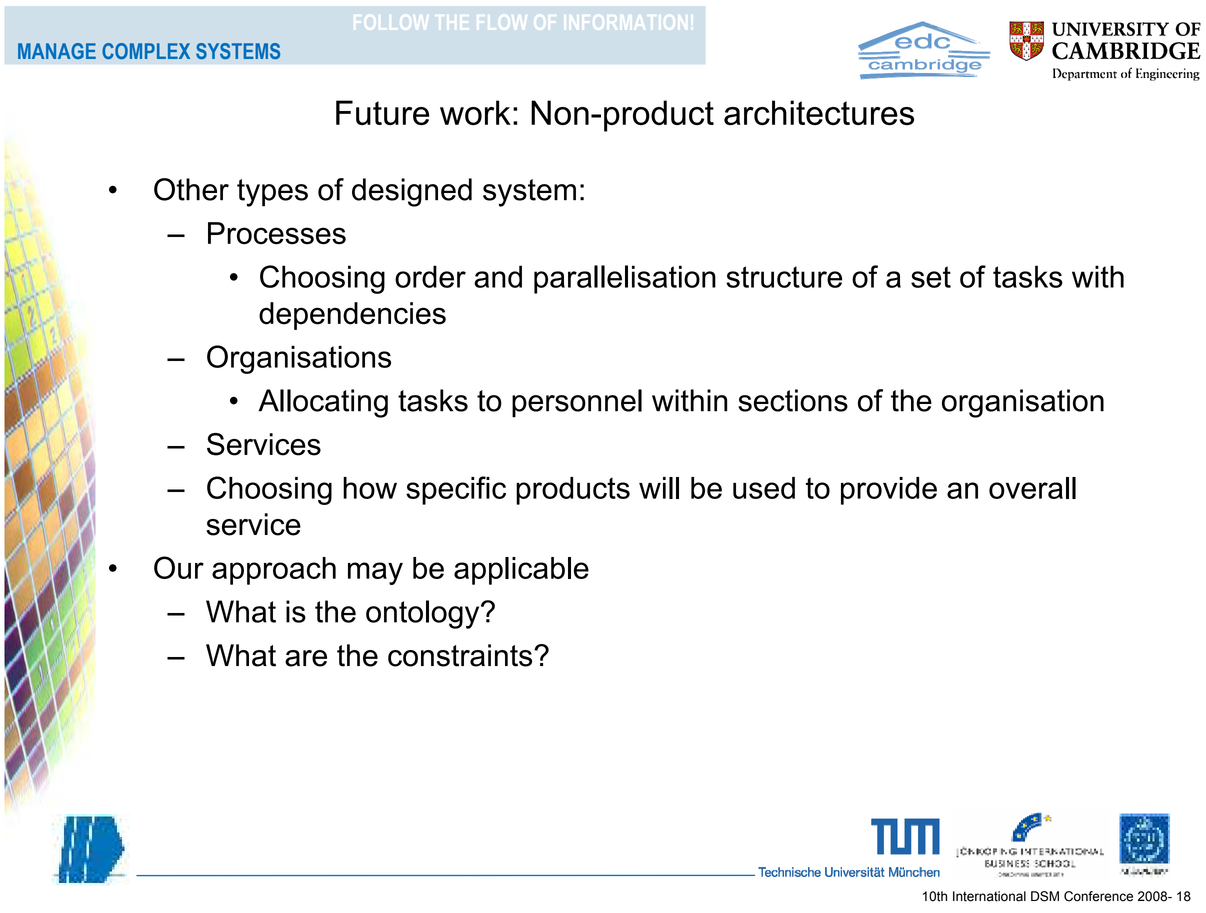


Future work: Design-by-redesign





- Instead of starting from blank DSM, 2 possible ways to incorporate elements of existing product architectures:
  - Manually specify subsections to redesign
    - Straightforward
    - May be difficult to identify most important section
  - Computationally explore possible "mutations"
    - Allows speculative "what could we gain" investigations
    - May produce infeasible suggestions
- Can be used separately or together
- Also allow <u>assessment</u> of existing architectures by comparing with "near" alternatives



Future work: Non-product architectures

- Other types of designed system:
  - Processes
    - Choosing order and parallelisation structure of a set of tasks with dependencies
  - Organisations
    - Allocating tasks to personnel within sections of the organisation
  - Services
  - Choosing how specific products will be used to provide an overall service
- Our approach may be applicable
  - What is the ontology?
  - What are the constraints?

Summary & conclusions

- Computational architecture synthesis can help produce better designs and improve existing ones.
- A formalism for establishing the validity of component DSMs is the Architecture Schema presented here, consisting of an ontology of components and connections and two types of constraint (connection requirements and path requirements).
- An exhaustive search algorithm can use such a schema to generate all possible architectures allowed by the constraints.
- Further work will be undertaken on applying this method to problems involving the redesign of existing products and to systems in other domains (processes and organisations).

Appendix: architecture evaluation formulae

$$designability = e^{Skewness(z_i)} = e^{\left(\frac{\sqrt{n(n-1)}}{n-2} \frac{\frac{1}{n}\sum_{i=1}^{n}(z_i - \bar{z})^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(z_i - \bar{z})^2\right)^{3/2}} \right)}$$

$$changeability = Mean(d_{ij}) = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n} d_{ij}}{\frac{n(n-1)}{2}}$$

Where:
- n = total number of components
- d_{ij} = shortest graph distance from component i to component j
- z_i = number of connections of component i
- $\bar{z}$ = mean number of connections per component

$$= \frac{\sum_{i=1}^{n} z_i}{n}$$