Schemebuilder

concept presumes that deep knowledge about design components has been acquired and represented in a preconceived way (information about it must be stored in the CLIPS database), and that there is a much codified insight into how the components operate in conjunction and how this operation can be evaluated

conceptual design only in so far as deals with choice of components in system at a high-level of abstraction, avoiding the detailed design

benefits of the approach: depends on the extent to which the system can generate options for system configuration that are acceptable to the human designer (automation of choice presumes that the options identified by the computer would genuinely be candidate designs for an intelligent designer, and makes most sense where there are a very wide range of possibilities that couldn't easily be surveyed and evaluated by the designer). Possible criteria to use to assess this: would like to think that a very large proportion of the suggested system configurations would be the configuration of choice in some circumstances, and/or that the computer can accelerate the evaluation process. Perhaps a good way to express the decision process is in terms of two phases: choose the system decomposition, by selecting components etc, then choose parameters to optimise the components to suit this decomposition.

usefulness of the system depends heavily upon being able to identify an area of application where it appropriate to use the design paradigm of choosing components from a catalogue to a template, then proceeding to detailed design. presumes that the kind of analysis that is needed to populate the database for Schemebuilder has first been done: not clear whether this kind of analysis is typical of existing design analyses - if not, then adds considerable investment overhead. also need to have confidence that the implementation platform is sufficiently stable to justify the investment (the evolution from KEE, Matlab and Simulink sends disturbing signal there)

good points of Schemebuilder as far as I can judge: bond graph concept provides very useful conceptual framework, lends itself to the analysis of system dynamics in traditional mathematical modelling idiom. functionally-oriented, in so far as the function of a system can be captured by such models. particularly valuable in this respect, since it redresses the balance between geometric and functional design, and suggests a process that in some applications will be better conceived than starting from the geometry etc. also draws on one traditional mode of design that invites automation

limitation: seem to be two dimensions to design
- breadth - want to study many alternative designs
- depth - want to examine and develop a particular design in detail

idea of a preconceived framework for choice and evaluation of a design is best suited to a process-oriented perspective. Works well only if we know what kinds of decision must be taken at each stage, and can anticipate the interactions in the design process. not clear that Schemebuilder is well-suited to handling interaction between different design views, since appears to be based on a formulaic systematic consideration of design concerns rather than integrated concurrent development from several design viewpoints

problematic aspect: danger that the automated design concept is taken too far. For instance, idea of encompassing geometry looks dubious (cf problems of beginning with geometry and embracing function). Approach can only be valuable in this aspect if the geometrical constraints and expectations are very modest. Important issue is to determine where the boundaries must be set for the Schemebuilder approach: to identify where it might be an appropriate design tool, and where its concepts are not applicable.

Conjecture: not good for aesthetic design, or where dominant constraints are imposed by context for use (e.g. size, shape, environmental factors?), mode of use (e.g. suitable for blind person, fault tolerant?). Idea here is that - whilst it may in general be possible in principle to

develop enough experience of such issues that it becomes possible to capture relevant knowledge in the underlying database - many designs pose one-off specific challenges for which it is not cost-effective to seek a general solution, whilst others involve considerations in which human judgement and idiosyncrasies are critical.

Schemebuilder can be seen as supporting a spreadsheet-style paradigm for design, where declare the structure of the system, then experiment to determine best choice of parameters, but need to be in situation where the framework for the experimental process has been predetermined and is no longer open to further elaboration. Particular issue for the proposed project is whether tolerances are an area where this kind of experimental insight exists: in principle, we can use Schemebuilder to record the interaction between "choice of parameters within given tolerance" and "system performance", but this may be a difficult relationship to capture in a closed rule-base in general. need a theory for tolerances.

Overall impression: Schemebuilder relies heavily in many aspects on preprocessing of design knowledge to a very high level of sophistication - this poses challenges both for analysis of designs and for choice of application area. Can't do much to change this state of affairs, so as to achieve greater flexibility, without change of modelling paradigm. Current software components have capability to deliver good performance where the knowledge representation / theory building task can be / has been accomplished thoroughly, but are not well-suited to the flexibility that is required if the design process is less precisely identified, involves more essential human judgement, and involves a high degree of interaction and iteration between different design viewpoints. Possible strategy: attempt to emulate Schemebuilder using Empirical Modeling tools (comparative study?)