6

Integrated Platform for AI Support of Complex Product Design

J.E.E. Sharpe
Engineering Design Centre, Lancaster University,
Lancaster LA1 4YR, United Kingdom
tel: +44 1524 593053; fax: +44 1524 593042
email: edc@comp.lancs.ac.uk

Abstract

Engineering Design is a highly integrated and integrating process. This paper describes an integrated computer based platform for the development from first principles of design solutions to appropriately defined need, and the choice and quantification of the best forms of embodiment.

The first part concentrates on the use of AI techniques in the rigorous development of functions, means and embodiments able to meet the prescribed functional need within the defined context. Details are given of the structure of the underlying knowledge bases and the bond graph based ontology. Spatial reasoning is discussed as part of the Function Means development along with a description of the use of working principles in the development of conceptual designs.

The second part concentrates on the use of mathematical techniques based on sophisticated simulation and symbolic manipulation in the development and quantification of the conceptual schemes generated by the use of Artificial Intelligence. The methods of selecting the best embodiments of the required functions are described, together with the means of determining the best size of component. The use of function maps relating function to cost, weight, limits on strength, efficiency of operation etc., are shown. Finally, details are given of the use of automatically generated simulations of chosen embodied schemes. These simulations are used to confirm expected performance and as part of the procedure for optimising the design.

Keywords

Conceptual Design, computer support, Artificial Intelligence, Function Means, embodiments, working principles, knowledge bases.

Professor Sharpe has now retired and may be contacted at 109 Copers Cope Road, Beckenham, Kent BR3 1NY, UK. Tel: +44 (0) 181 658 9271

T. Tomiyama et al. (eds.), *Knowledge Intensive CAD*

1 INTRODUCTION

The engineering design process typically entails the construction of a description of an artefact that satisfies a functional specification, meets certain performance requirements and resource constraints, is realisable in a target technology, satisfies one or more criteria for manufacturability: reliability, safety and more recently environmental constraint. The common perception of engineering design is often taken as that of converting a need - expressed as an abstract concept in terms of general functionality - into a product fulfilling that need, and its process involving the mapping of a specified function onto a realisable physical structure - the designed product.

The engineering design process is a complex one that requires the design engineer to exercise initiative and inventiveness as well as deploy a wide range of skills and expertise in attaining a solution. In an interdisciplinary design environment the designer is also often required be a generalist with an eye on the possibilities of using a wide range of technologies. The increasing occurrence of interdisciplinary product development has not only removed many of the traditional constraints to design but has now given the designer a much wider freedom of choice as to the best solution to a particular design problem.

Currently, much of engineering design is carried out by individuals or groups of individuals who have been trained in one engineering discipline with limited knowledge or experience of others. These individuals "lock onto" traditional and familiar technologies that may result in a less than optimal solution to their design problem. There are two ways to interdisciplinary product design: (1) educate all designers with a generalists' knowledge of many technologies, and (2) co-locate technology specialists to encourage communication and co-operation in the design team. However, with the advances made in computer-aided design and the infusion of AI outgrowths like functional reasoning, constraint processing and planning into traditional computer-based design tools, it is not difficult to see how a third alternative to the above approaches might arise - develop a computer system to support and guide the designer through the range of technological options available, aided with the appropriate tools to perform processing of the design data at the proper level for design abstraction, concept generation, technology selection and matching, model execution, optimisation, and visualisation.

The aim is to research the design methodology required and embed this into such a system. A set of tools that provides highly integrated support for the rapid creation and evaluation of a wide range of outline schemes incorporating a range of technologies and to enable the comparison of technological alternatives to take place before large commitments are made and irrevocable decisions taken on the basis of biased information has been demonstrated.

2 AIMS OF THE COMPUTER AIDED TOOLS

The main remit is to assist the designer in the conceptual and embodiment stages of design, including problem analysis, of interdisciplinary systems. The design of mechatronic systems, which combines mechanical, electrical, electronic and software systems, presents a challenging array of possible implementations, often based on proprietary equipment but with opportunities for elegant and original design solutions that require the deployment of advanced electronics and software techniques to provide the required levels of functionality. To take full advantage of these opportunities requires the ability to generate and evaluate quickly alternative schemes from first principles to practical embodiment.

The system is a kind of "design workbench", where designers are guided through the range of technological options available, aided by the provision of tools for rapidly accessing relevant information. It facilitates the creation of a model of the system to be designed, and advises on the

appropriate means to achieve the design and the exploration of alternative conceptual schemes with an appropriate allocation of function between mechanical, electronic and software elements.

The definition of a scheme follows that of French (French 1985) *"a scheme is an outline of a solution to a design problem, carried to a point where the means of performing each major function has been fixed, as have the spatial and structural relationships of the principal components. A scheme should be sufficiently worked out in detail for it to be possible to supply approximate costs, weights, and overall dimensions, and the feasibility should have been assured as far as circumstances allow. A scheme should be relatively explicit about special features or components but need not go into much detail...."*

The philosophy underlying the work is not that of automated design using the expert systems approach (Rychener 1988), but one embracing a more co-operative relationship between man and machine as underlined by the work of Fischer (Fischer 1990) and described by Oh (Oh 1993) in the belief that a more appropriate approach to computer-based design tools is for the computer to provide decision support and allow the human designer to apply the judgement.

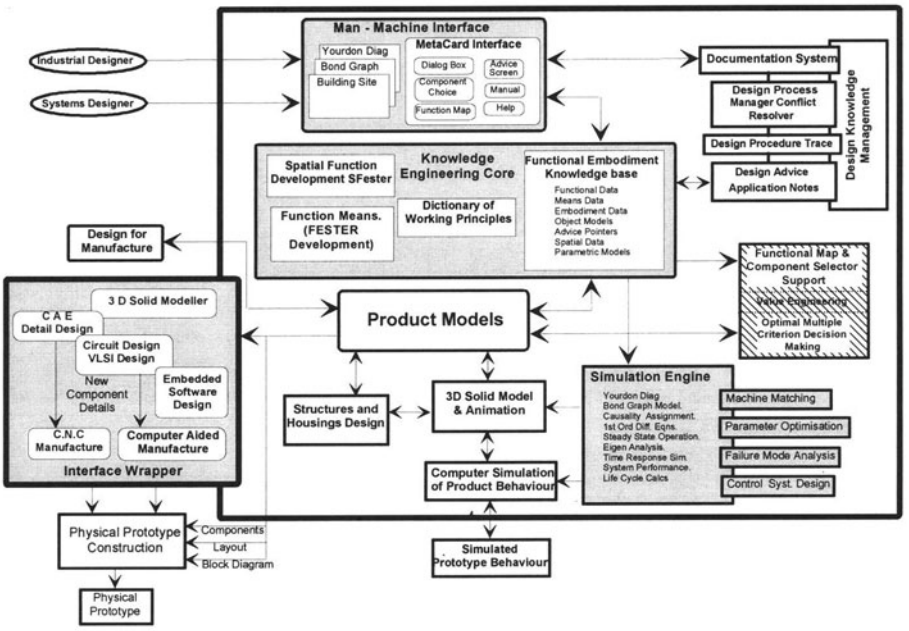

Figure 1 Overall architecture of the Schemebuilder environment.

2.1 The Support Environment

An integrated assortment of design tools matched according to the specific needs of different design tasks, including the provision of verification tools like those for simulation in the Schemebuilder environment (see Figure 1), helps to tighten the link between design and analysis and enables designers to predict the behaviour and performance of their digital "mock-ups" as early as possible. In short Schemebuilder allows for the exploration of new ideas with minimal risks. Tools are provided to:

- assist in the production of specifications
- provide advice and browsing facilities on available technologies
- aid in the identification of "technological holes"
- produce system models for dynamic simulation and parameter optimisation
- support embedded software design and development
- monitor system integrity, checking for continuity and matching
- produce preliminary layouts in the form of realistically sized "solid sketches"
- design and model casings and support structures

2.2 Bond Graphs

As products become more complex and highly integrated, it is increasingly necessary to have a common language in which to communicate. The use of bond graphs (Paynter 1961), (Karnopp et al. 1990) and (Sharpe 1978) with its common structure and clear rules across all the engineering domains for the representation of the functional and behavioural aspects of energetic systems, allows a very natural approach to design and the development of an integrated suite of objects and rule-based computer aided conceptual design tools.

Bond graphs are a formal language for representing physical systems. They have been used quite successfully in the creation of formal models for mechanical design, for example (Finger and Rinderle 1989) and (Ulrich and Seering 1989). There has also been an increasing interest in the use of bond graphs for qualitative modelling in AI because of their simplicity and representational power (Fishwick 1989; Soderman and Stromberg 1991; Top and Akkermans 1991).

The modelling process is based upon the conservation of energy, with physical processes being linked in a labelled digraph through energy flows. Basic concepts like effort, flow, inertia, and capacitance are used in modelling, and their generality allows the method to be applied across domains like thermodynamics, rotational and translational mechanics, fluid dynamics and electronics. The bond graph can also support a useful set of qualitative inferences including an analysis of the causal ordering that occurs within a system.

3 SCHEME GENERATION CONCEPTUAL STAGE

The Schemebuilder design workbench described previously in (Bracewell et al. 1993; Sharpe and Bracewell 1993) has been enhanced so that schemes are developed using a variant of the Function Means Tree approach of Buur (Buur 1990), guided by a structural set of design rules and working principles. The Function Means Tree approach, aided by the application of stored, rule-based conceptual design principles, allows for the progressive refinement and development of a design from required functions expressed abstractly to the eventual physical embodiment of those desired functions. Backed by an Assumption Truth Maintenance System (Filman 1988), it is also possible for the tree structure to maintain the development of all alternative schemes, even though the designer may only progress a limited number of them at any one time. Figure 2 illustrates conceptually the process of generating from a statement of need a set of alternative schemes, and the assessment of these schemes.

Although it is important to understand the mechanism whereby a design is developed from first principles, the development of a conceptual design may begin at a number of different starting points. Any satisfactory computer aided procedure must allow for this.

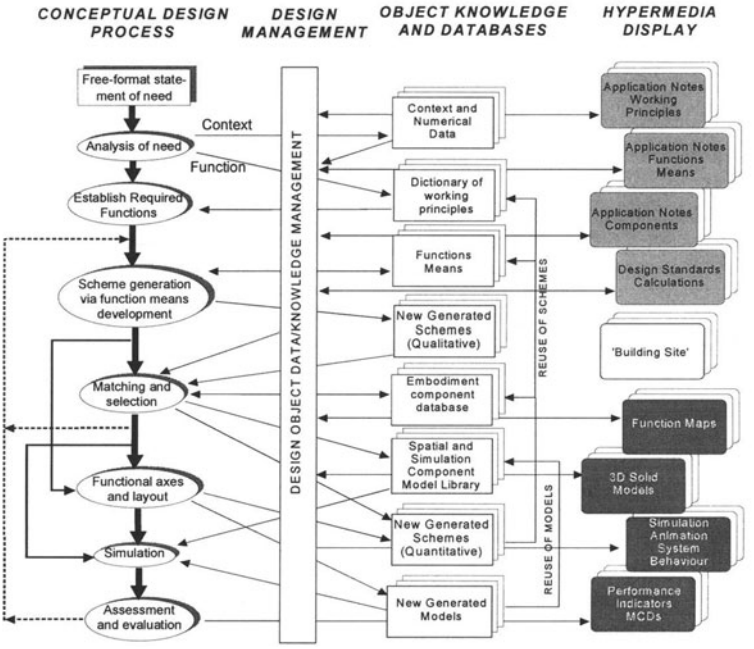

Figure 2 Schemebuilder conceptual design process.

3.1 Scheme Generated from Experience

The designer may choose from past experience of a satisfactory similar design to use an established design as a starting point. Such a scheme may be laid out on the *Building Site* as a set of interconnected blocks representing existing types of components. The primary functions of those blocks together form an established working principle, with each component represented by a block, being an embodiment of some working principle which then further defines another set of primary functions for that component. Figure 3 shows such a block diagram for a metabolyte infuser, which will be our illustrated design exemplar in this paper.

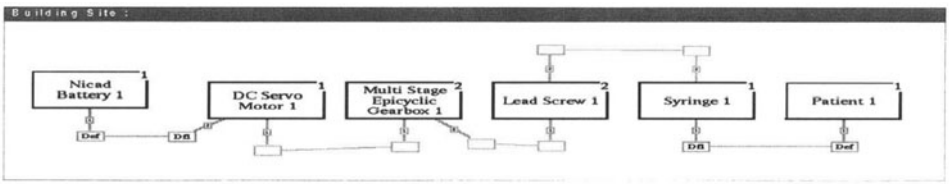

Figure 3 Block diagram of metabolyte infuser.

It may be of no consequence to the designer to know the nature of the primary function performed by the design. The designer's concern is that the concept works and that it may be matched to the desired specification and designed in detail.

3.2 Scheme Generation from First Principles

Frequently however, the designer wishes to develop a conceptual design from first principles, exploring the myriad of alternative schemes that are physically possible before choosing the best alternatives to pursue into detail. Such a process will begin with some "expression of need" combining normal language and related numerical information. For example, the "expression of need" for the metabolyte infuser could be defined as follows:

"The need is for a light weight portable infuser capable of providing a continuous constant flow of drug into a patient's artery for up to 48 hours. It must provide an indication of correct functioning. The flow rate is to be 2.5ml per day."

Employing techniques from AI such as natural language processing and information retrieval in the implementation and processing of dictionaries, thesauruses and grammar, it is possible with the help of the designer, by means of prompts and iteration, to establish in Yourdon-like terms (Yourdon 1989), the context of the design, and the desired function. Furthermore, it is possible to use a thesaurus to search a dictionary of working principles to match those words appearing in the functional aspect of the desired need with the key words describing the application of one or more principles.

3.3 The FESTER, Functions, Means and Working Principles

The *FEST-ER* stands for **F**unctional **E**mbodiment **ST**ructure - **E**xtended **R**ecursively. It is a new information structure extending the capabilities of the Function-Means Trees of Andreasen (Andreasen 1980). The FEST-ER is developed by the designer's free choice of stored functional embodiments, which may be of either two forms: means or principles.

Nomenclature Definition

A *function* can be classified as either a *data function* or an *energy function.* All functions are pre-defined in the Function Embodiment Knowledge-base. Functions may have one or more defined *attributes*. Functions communicate with each other and with components via *links*.

A *link* may carry energy and/or data. In a completed scheme both ends of every link must be connected to a component within the scheme. The link may traverse up and down a number of levels within the FEST-ER between its terminating components. Each link has a unique identification.

An *attribute* more precisely specifies the action of a function.

A *functional embodiment* can consist of either a means or a principle.

A *means* consists of at least one component and, if necessary, one or more required subfunctions which may have certain required attributes. Links joining components and required subfunctions within the means must be defined.

A *principle* consists of one or more required subfunctions which may have certain required *attributes*. *Links* joining required subfunctions within the *principle* must be defined.

A *component* represents a family of actual available physical items, for which quantitative information is stored in the Component Database, allowing sizing, matching layout and simulation to be performed later in the design process. Components possess ports which form the end-points of links, allowing input and output of energy and/or data from the component.

3.4 The Functional Embodiment Knowledge-Base

The Functional Embodiment Knowledge-base is a tree structure of functions in which energy flow types and data carrier types are progressively defined as you approach the levels of the tree. This allows principles to be stored at a high level of abstraction with energy types undefined, then

inherited into all appropriate energy regimes. A crucial aspect of the structure is the ability of single component types from the component database to appear in multiple means, embodying different functions at different points in the tree. This reflects the fact that components are often used in a number of alternative and sometimes unusual ways.

3.5 Working Principles and their Application

Consider the case of the *need* for a flow of chilled water at $5^{o}C$ for the purpose of cooling power electronics. The flow is to be 1 kg per second within a closed circuit. The return water temperature is $15^{o}C$. Power is to be taken from a variable frequency voltage source.

Considering the word "chilled", a search of the Dictionary of Working Principles identifies a group of Working Principles relating to refrigeration (refer to Figure 4). These include: Air Standard Cycle; Absorption Cycle; Vapour Recompression; Evaporation; Seebeck Effect; Peltier Effect and the Joule-Thompson Cooler. Each of these alternatives of working principles has a required set of primary functions that may be satisfied by different means and these functions, together with their interconnections, are held in a related database. Thus, for the chosen principle to be embodied, the functions are drawn on the highest level of the building site and placed at the head of the FEST-ER together with its context, in this case a closed circuit and the high temperature product stream.

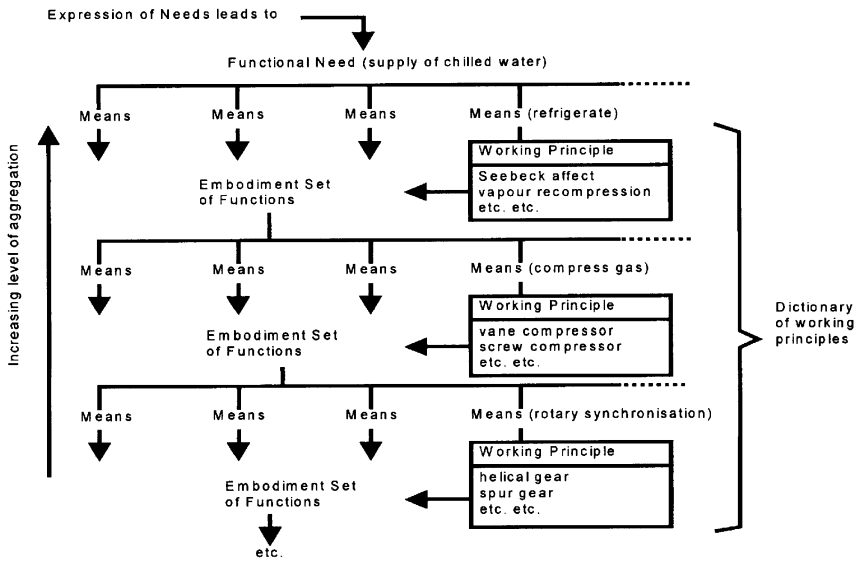

Figure 4 Working principles in qualitative development of a scheme.

For the refrigeration application, the choice of the best working principle is not clear, and therefore requires recourse to background information on the application of the different principles. This is provided through on-line hypertext links to appropriate handbooks or application notes and functional performance maps held in MetaCard, a hypermedia development environment.

Having chosen one or more probable applicable working principles, it is necessary to determine the means of achieving the required functions. This is performed by searching the available means to achieve the function or alternatively decomposing the required function according to a set of rules,

the Rules of Decomposition, and searching for available means of performing the generated sub-functions.

Each means may take a number of different physical forms depending on the working principle employed. Thus we find that if we choose the vapour recompression cycle embodiment of the required refrigeration means, the gas compression function may be achieved by different means and each of these means may itself have different embodiments depending on the working principle used. This may be a Reciprocating Piston, Rotary Vane, Screw Type or one of many others.

Suppose the designer chooses the Screw type; this will require a number of functions, one of which will be to synchronise the rotation of the screw elements. There will be different means of achieving this, such as Gear Means, Toothed Belts or Electronically Phase Locked Rotation. For each of these means there will be alternative embodiments. The gears may be helical or spur and will in turn lead to a further set of required functions which in turn lead to further required means and embodiments of working principles.

In this example there are some four levels of working principles, as indicated in Figure 4. For a complex product there may be seven or eight levels of working principles and their embodiments, each requiring its own function means tree, giving rise to a multiple level hierarchy of Means, Principles, Embodiments and Functions. Spatial aspects of the functions, links to the environment and the need to provide physical structure, may occur at any level in the hierarchy. Thus spatial principles of structures and kinematics may be invoked at any point. The starting part of the conceptual design decomposition is not restricted to any particular level of aggregation.

3.6 Bond Graph-based Principles

Bond graphs (Karnopp, Margolis et al. 1990) provide a unique set of principles for the embodiment of energetic systems. These principles are based on the facts that only compatible energy ports may be connected to each other, that the causality and the dependency of the energy covariables must be correctly propagated and that there are a limited set of primary functions that may be used. These include the transformer (TF), the Gyrator (GY), the REsistor (R), the Compliant (C) and Inertial (I) energy stores, the Common Effort (0) and Common Flow (1) Junctions and Sources of Flow (SF) and Effort (SE). Each has a clear set of principles governing its function and how it may be decomposed whilst still performing the same overall primary function (Sharpe 1978).

The following is a partial list of the procedure for applying the Decomposition rules:

1 Search Function Means Database for Any Known Means; if none
2 Insert Transformer to provide intermediate energy domain; or
3 Replace Source of Effort with a Compliant Store; or Source of Flow + Conflict Resolver[CRES] ; or
4 Replace Source of Flow with an Inertial Store; or Source of Effort + CRES or
5 Replace Inertial Store with Compliant Store + CRES; or vice versa.

The decomposition may be taken to any level, although to go beyond 4 stages of decomposition is normally unnecessary within system design. The ability to split a function and introduce an intermediate energy domain allows the coupling of domains that are not normally possible. For example, it is not possible to transfer energy directly between the electrical domain and the incompressible fluid domain, which will be illustrated in the metabolite infuser example where it is necessary to use either a translational or rotational domain as an intermediary.

The FEST-ER is complete when the designer is satisfied that there are a suitable number of complete schemes of embodied means to choose from and to take into the next phase of the design process, the component choice, simulation and layout.

3.7 The FEST-ER

The FEST-ER differs from the well-known Function-Means Tree in that it supports two forms of non-hierarchical links that are very important for the efficient development and representation of competitive designs.

The first of these is the Embodiment Function Reference, which is used when a function, which had already been embodied at some point in the FEST-ER, occurs again in a location which is non-mergeable with the original occurrence. Instead of embodying the function all over again, an Embodiment Function Reference can be made, reusing the original work. There are many examples of this in Figure 6, where reference functions are surrounded by heavy boxes and references to them by light ones.

The second form of non-hierarchical link occurs where a single means is, when appropriate, allowed to embody more than one function, which can often be the key to a more competitive design. This technique is also the key to representing ASIC hardware code or embedded microprocessor software means, for implementing data processing functions. Once this mapping has been made, FEST-ER contains a complete Yourdon-style structured analysis of the information processing required of the microprocessor or ASIC.

If the individual data functions used are stored in the Functional Embodiment Knowledge-base with their representation as High-level Petri Net fragments (Jensen and Rozenberg 1991), then a complete net describing the desired functionality can be automatically assembled. The resulting net, if exported to the commercial software tool "SystemSpecs", will then allow automatic generation of simulation or implementation codes in C++, Occam or VHDL (IvyTeam 1993).

The FEST-ER information structure will equally support alternative embodiments of the same functionality in, say, analogue electronics or even a special mechanical linkage, thus allowing trade-offs of moving solutions either way across the "mechatronic interface" to be easily examined.

4 INITIAL FUNCTION-MEANS DEVELOPMENT

Having shown how the function means embodiment decomposition fits into a hierarchy of working principles and the features of the FEST-ER, attention will be focused on the development of a FEST-ER for the metabolite infuser example, which only requires one level of aggregation. This example may be decomposed into its alternative constituent components from a single functional requirement, a constant source of flow of an incompressible fluid, shown diagrammatically as SF.IFLU, within one level of the hierarchy Figure 5.

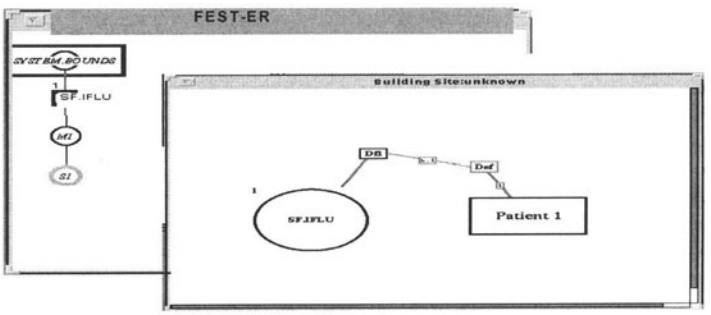

Figure 5 Initial development of the FEST-ER

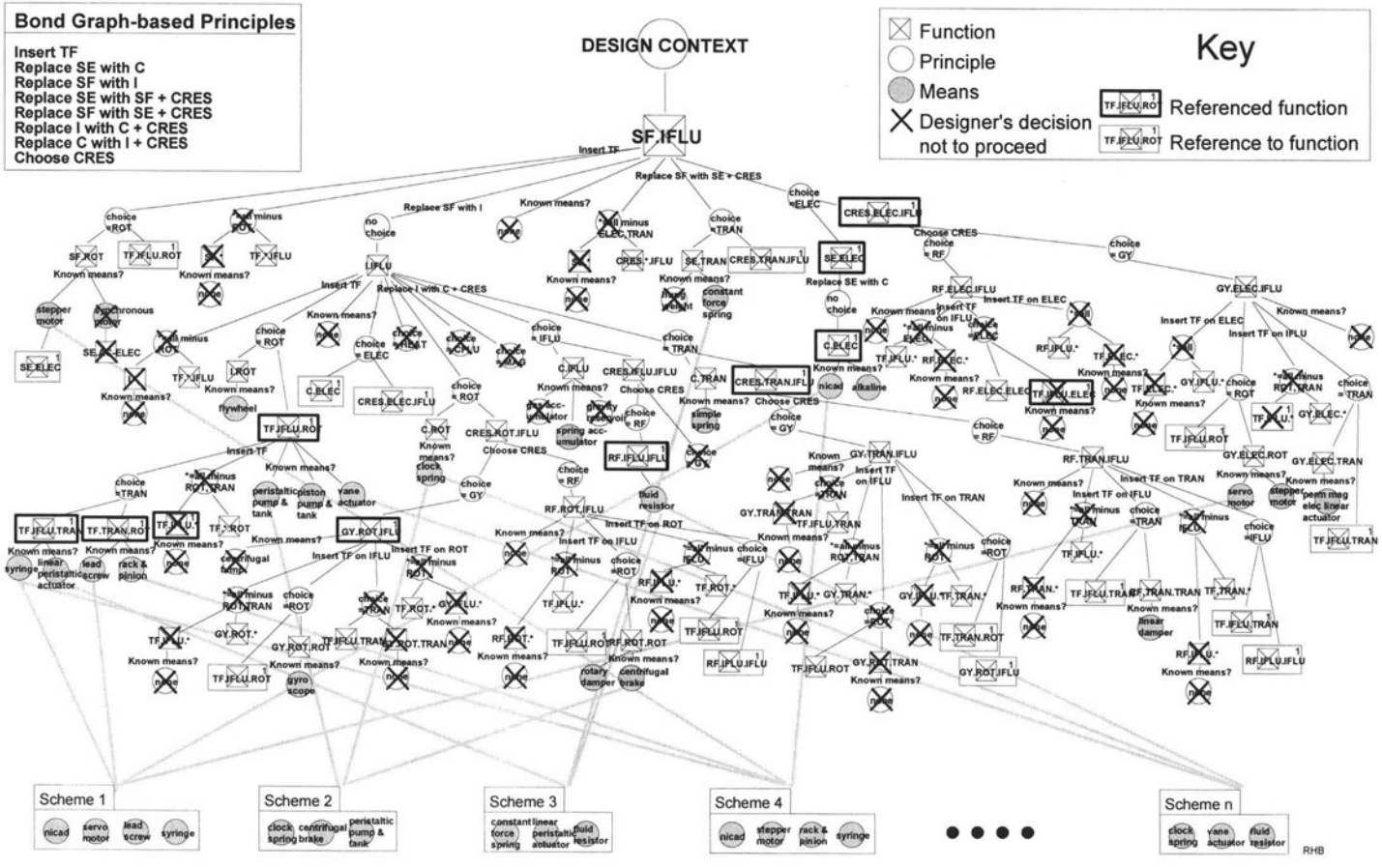

Figure 6 The FEST-ER of Metabolyte Infuser showing several of the schemes generated

The use of Bond Graph reasoning provides a unique set of rules for the decomposition of energetic systems. These rules are based on the facts that only compatible energy ports may be connected to each other, that the causality and the dependency of the energy covariables must be correctly propagated and that there are a limited set of primary functions that may be used.

4.1 The Complete FEST-ER for Metabolite Infuser Example

A good example of the use of bond graph-based principles is the Metabolite Infuser. The FEST-ER for the metabolyte infuser is shown in Figure 6. This shows the total development of the alternative designs including the points at which the designer has chosen not to proceed. Any of these branches may be revisited at any time if required. The procedure begins with the definition from the expression of need of the desired function and its context. This is shown diagrammatically at the top of the FEST-ER structure.

In this case the required function is a Source of Flow in the incompressible fluid domain SF.IFLU. The application of the first rule to search the Function Embodiment Knowledge-base returns "no known means". The insertion of a Transformer [TF] to give an alternative intermediate energy domain shows that for the intermediate rotation there are several alternative transformers, from rotary motion to incompressible fluid flow TF.IFLU.ROT, but that there are no matching sources of Rotation. Replacing the Source of Flow by an Inertial Energy Store yields "no known means". Replacing the Source of Flow by a Source of Effort or Compliant Store and a Conflict Resolver, which may be a Resistor or a Gyrator, yields a whole array of possible alternative energy domains and means. The alternative energy domains are electrical, rotation and translation, two stages of rotation, or two stages of translation.

Each transformation or gyration between the domains has several different means, which may give rise to some 20 alternative embodiments that have the potential to perform the desired primary function. Each represents a working principle, and as such may be added to the dictionary of working principles. Each scheme represents the minimum functional structure and is at this stage purely qualitative.

4.2 Iterative Relationships of Function Means Development

The schemes presented from the Function Means decomposition represent assemblies of embodied means and therefore are automatically represented in bond graph form. They do not contain any knowledge about the spatial arrangement of the designs unless it has been specifically included in the definition of the desired function. As the chosen scheme is laid out in the 3D solid model it will be necessary to introduce additional functions, principally spatial transformers, to couple the differing functional axes (offset axes X1 and X2) of the primary functional components. This will introduce additional functional needs into the chosen scheme as shown in Figure 7, in which the primary functional components and the initial context form the context for the development of the function means for the additional coupler.

The application of the decomposition rules leads to the choice of a cantilever slide or a pivoted lever. With the choice of the cantilever the new modified scheme may be laid out in the building site. The efficient matching of the DC motor to the load requires the introduction of a further rotary transformation with common axes (X1) for input and output. A search reveals that the epicyclical and the multistage gearboxes are suitable means. The gear pair is not suitable because the input and output axes are offset. Thus, there is a continued development of the FEST-ER and the alternative schemes as the detail matching and layout of the chosen schemes proceed. The bond graph models generated for each scheme provide the basis for the simulation to be described later, and together with the defined functional axes, allow the forces and torques within the functioning scheme to be superimposed on the 3D solid sketch of the design.

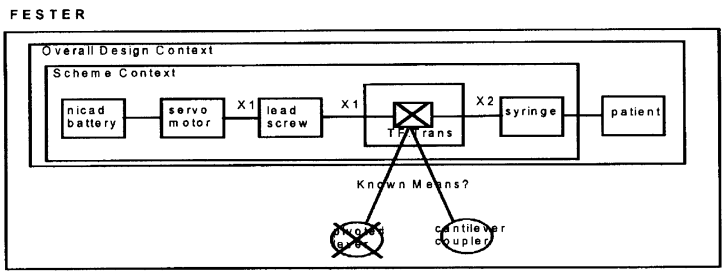

Figure 7 Function means interaction with functional axis.

5 THE EMBODIMENT DESIGN OF MECHATRONIC PRODUCT SCHEMES

The embodiment design is characterised by achieving the task of quantifying design parameters so that an optimum design scheme is obtained to fulfil the embodiment of a working principle from which a particular scheme is derived. It is very often necessary to evaluate several schemes or variants before a designer can make sensible judgement and comparison. Embodiment design therefore involves various aspects of specification and evaluation; and several computer aided supports are needed to enable a designer to carry out embodiment design effectively. These requirements have been provided for by supporting tools that come under four major categories; component function mapping and matching, data serving, simulation and 3-D modelling and layout design tools, each of which is described in this paper.

5.1 Facilities For Quantitative Definition Of Schemes Embodiment Stage

The quantitative definition of schemes corresponds to the embodiment stage of design. The main facilities involved in enabling the designer to numerically quantify the schemes generated from the previous stage are discussed in the next few sub-sections. The typical processes involved at this stage are: (1) taking a product, component by component, at any stage in the overall Schemebuilder process and sizing them or selecting them so that their individual numerical data is defined sufficiently as to allow realistic simulation of that product; (2) production of layout drawings with leading dimensions given, allowing estimates of cost and weight to be made. On the basis of these measures the designer is able to select the most suitable scheme or scheme-variant.

6 DATASERVER

Each design primitive in a scheme is represented as a generic component object, which is an instance of a class of components that can physically realise the required function. If a generic dc-servomotor object, say, "dc.servomotor.1" is the design primitive in a particular scheme, then "dc.servomotor.1" is an instantiation of the DC.SERVOMOTOR class and it represents the means by which to satisfy the required function "to convert electrical energy to rotary motion". Although "dc.servomotor.1" is in truth an instance of the DC.SERVOMOTOR class, it is not unique in the sense that it does not have specific values for its structural attributes, performance attributes, and so forth; "dc.servomotor.1" is

not a specific instance. It is not a distinguishable object at this point in time and hence does not have a direct mapping between itself and the physically available object in the real world.

Each component class has an attached object called a *dataserver*, which acts as a data-providing agent to furnish the generic component object with real values in order to specify the object fully. Each dataserver contains a schema descriptor, parametric formulae, structural, functional, performance and cost data related to the component it is attached to. The structural, functional, performance and cost data are provided by a database which is loaded at run-time and only when the dataserver is called upon to supply data. The database, structured as a one dimensional relational structure, stores component information taken from a manufacturer's catalogue and values for bond graph constants.

The schema descriptor provides the mapping of labels to columns, e.g. "model.type" to column 1, "mechanical.friction" to column 2, "diameter" to column 10, and so on. It also provides further information like units that are associated with a particular attribute, e.g. "mNm" for mechanical friction. Mapping of attribute labels to parametric formulae is also provided by the schema descriptor.

When a particular model type is chosen, for example from the Matching and Selection tool, all information along with the pointer to that model, that is needed to specify fully the nature of the design component is transferred across to the generic component object. The generic component object is now said to be fully specified. The information can then be used to draw the physical representation of the component onto the Layout tool and provide more detailed information for simulation purposes. Some of the information transferred across to the component object is also taken up by the bond graph models.

7 COMPONENT SPECIFICATION MATCHING

The facilities for matching implemented within Schemebuilder allow the user to proceed to a scheme which is sufficiently numerically defined to allow assessment by means of simulations and trial layouts. The facilities described below can be used in any sequence but are described in that of normal use. Tools associated with the Dataserver allow the user to select candidate components from the database of a component type such that they meet an inclusion set of performance-related requirements. The tool allows the candidate components to be listed in order of least cost or weight, for example. In general the components are retrieved via SQL-like querying functions using any of the descriptive parameters in the database of that component. Having selected sets of components as described above their respective suitabilities can now be examined. The graphic tool for matching displays the performance envelopes of the candidate components together with the working points derived from the Numerical Specifications. These working points represent a number of salient conditions. The position of these points on the respective envelopes of the candidate components allows the user to judge their suitability and make a selection.

The process described above is repeated at each component boundary to propagate the specification requirements through the scheme. In order to do this, the requirement has to be carried through each component using the bond graph constants stored in the component database.

In general for a design to be economically viable, components have to be approximately matched in terms of power, with individual components operating in their most efficient and economic regions of their respective operating envelopes. However, in the matching processes described above this may prove impossible unless a transformer device is interposed, e.g. the gearbox between the leadscrew and the motor in the example case. The user will be advised to introduce these elements when such conflicts arise, either by consulting the Design Advice or by the system recognising and flagging gross mismatches in power, effort or flow capacity between neighbouring components.

7.1 Check calculations facility

Particular components often have specific features that must be checked before they can be accepted for a scheme. In these cases the user is offered advice to use such checks as part of the defining process. Specific check tools would include, in the case of lead screws for example, active formulae for the calculation of whirling speed; or in the case of an electric drive might demand the simulation of a motor in isolation, but under set conditions, and to assess its likely running temperature using the simulation tool described below.

7.2 Inter-Component and Inter-Scheme Comparisons

To further assist the user in choosing between components and schemes a facility is provided which allows comparisons on the bases of cost, weight, working temperature range, maintenance interval etc. The tool presents a list of comparison variables for that component type or scheme, and the user interactively selects items or combines them algebraically and displays the results as a series of bar graphs shown in Figure 8. Thus power per unit cost and per unit weight could be compared if a design of least cost and weight were desired. The tool uses data from component databases when doing inter-component comparisons and from scheme files when performing inter-scheme comparisons.

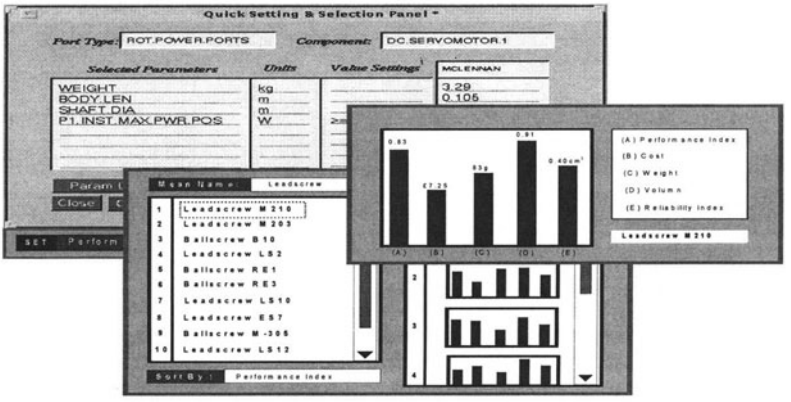

Figure 8 Component comparison tool.

8 SIMULATION FACILITIES

Bond graphs describe system energy variables through their bond elements by denoting effort variable flows in one direction and flow variable flows in the opposite direction which are decided by bond causal structure. It is therefore very easy to obtain the dynamic response of any energy variable from a bond graph simulation method.

The integrated workbench facility uses "Simulink", a block diagram based simulation software developed by the Math Works Inc. It is used to derive modelling and simulation methods for mechatronic products by selecting and aggregating high level functional modular blocks. A detailed dual effort and flow variable relationship block diagram model is created for each component from basic block elements. This is then built into a high level-of-function block which is created for each component type by grouping them into a single block and masking it to give an interface of a number

of inputs and output together with a dialogue window. These blocks can be collected into a library and are used to create models to simulate the dynamic performance of mechatronic products.

Each component type can be described with different levels of model complexity. Correspondingly different objects have been created and collected in the library to represent them. Up to three levels of model complexity have been considered for each component and these models are stored and marked for different levels of system performance simulation.

For mechatronic and other interdisciplinary systems, information system simulation can also be performed by Simulink. A number of elementary function blocks are created, and similar ones collected into libraries, such as a logic operation block library. These operation blocks can then be used to create a control device. Various sensors can also be created to form a sensors library which allows a user to create a model incorporating sensor devices. Sensor and controller libraries can similarly be created. All these sub-libraries together form a central library for modelling and simulating mechatronic products in a unified manner.

8.1 A simulation model example

The metabolyte infuser example is made up of a drive system comprising a battery, a D.C. motor, a gearbox, a lead screw with a special nut also acting as a coupler to push the syringe piston, a casing and a syringe. This can be modelled as simple function blocks with dialogue windows, where each component is modelled in a simple function block which in turn consists of a detailed dual effort and flow variable block diagram model. Each of these function blocks normally has a pair of effort and flow input variables, a pair of output variables and a component efficiency variable. Bond Graph Causality is used to determine which is the input variable between a pair of energy variables. Two ways of specifying parameter data are implemented, namely auto-data retrieval through an intermediary data file and user parameter specification through dialogue windows.

A graphic simulation output of motorised drug infuser system response to a step voltage input from the battery may be obtained and used to study the system's performance. Several simulation sessions can be conducted to evaluate the response of variant schemes within a product design.

9 LAYOUT FACILITIES

The Layout tool supports the preliminary embodiment phase of design, producing quick 3-D "solid sketches" of schemes. It provides a rough feel of how the components are to be assembled together giving a sense of their spatial layout. The intention is simply to "reserve space" in a design for final components, so the drawings can be much simpler than those which eventually replace them. Features such as fixing holes, spigots and fillets as found in detailed drawings are omitted.

The AutoCAD system has been used to accomplish the requirements of the Layout tool. The routines to enable the drawing of the 3-D solid representation of a component have been implemented using the AutoLISP macro language. The data necessary for providing values to the dimension variables describing the structural aspects of each component type is drawn from the set of values obtained from the dataserver. Each component is represented geometrically by a combination of 3-D solid modelling primitives, which include cylinders, boxes, wedges, spheres and cones. There are also procedures in which to produce solid models of more complex shapes via extrusion, sweeping, subtraction and union.

Figure 9 illustrates how the solid model of a typical product is represented. Components are represented as a union of primitives, the selection of either one depending on the values transferred across from the dataserver. The actual positioning of each solid primitive, with respect to each other, is determined by constraint relationships which are parametrically driven by the specifics of their structural properties. Hence the positioning of the "motor body" is constrained by the length of the

"motor shaft" which has a value supplied by the dataserver. The mating position, which is represented as a 3-D co-ordinate point, defines the point at which the next component in the scheme is to be assembled with the component in question. The functional axis defines the direction in which the next component is to be assembled for the valid functional interfacing between the two components. These two further aspects of representation augment the "vanilla" approach to solid modelling by allowing the system to assemble the components within a scheme automatically based on their functional interfacing as determined within the scheme representation. Based on the functional interfacing topology of the component models in a scheme, the Layout knowledge base generates a layout description of the assembly and sends it to AutoCAD to align and assemble the components automatically.

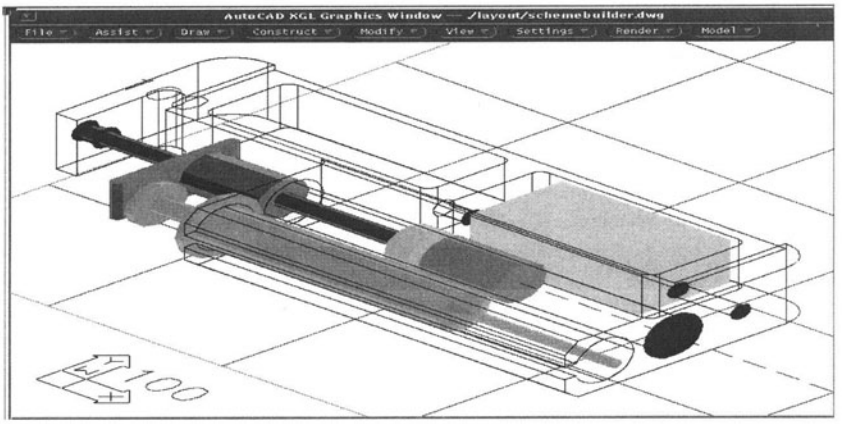

Figure 9 Solid model of a motorised drug infuser.

The process of drawing a component in AutoCAD in response to a triggered action in the AI system, KEE, is enabled by an agent called the Draw Manager. The Draw Manager is responsible for storing object representations of the solid primitives and, if necessary, stores the list of instructions needed to create a more complex solid model representation using a series of solid modelling operations like union, subtract, chamfer and fillet. Information is sent across to AutoCAD which will then execute the relevant AutoLISP procedures to draw the representation of the component.

Figure 9 shows the solid modelling layout of one scheme from a set of alternative schemes for the design of the motorised drug infuser. The components have been aligned and assembled automatically after execution of a spatial reasoning process activated when an option on the system menu is picked.

10 INTEGRATION MECHANISMS

There are essentially three integration methods employed in the Schemebuilder environment:

1) use of a unifying modelling technique, namely bond graphs, to integrate heterogeneous concepts in different energetic domains;
2) use of computational agents called design facilitators to map between data and knowledge structures and models in one system onto another;
3) use of the underlying operating system mechanisms - UNIX pipes and intermediary text files - to integrate the various software systems.

10.1 Design facilitators

Designers and engineers frequently carry out optimisation and simulation exercises, especially at the embodiment stage of design, to assess the viability of a chosen scheme. The tools that support such activities allow the designer or engineer to try out what-if scenarios and to rapidly change the values of design parameters in order to assess the effects of the changes. In interdisciplinary design, it is common to use a variety of heterogeneous sub-systems to model and analyse the different aspects of the design from different viewpoints. Design facilitators can help with the data translation between one tool and another, and in the browsing and retrieval of information from electronic catalogues and libraries. The design facilitator in Schemebuilder is an agent that provides mechanisms for the mapping and conversion of different schema and model representations from one tool to another.

The person responsible for constructing the schema description of the database of dc-servomotors for instance may describe the dimensions for the dc-servomotor as "body.len" (body length) and "body.dia" (body diameter). However, the knowledge base consisting of the classes of solid primitives may have the definitions different from that of the database. It is not practical and sometimes may even not be feasible to enforce congruency of the definitions in the database with that in the knowledge base. The design facilitator is an object wrapper that mediates between the two by providing a mapping mechanism to translate one schema definition to another.

The design facilitator also converts numerical values associated with conceptual notions used in one model to those in another model. In these cases the normal one-to-one correspondence between vocabularies does not hold. For example, the term "rotational resistance" and "mechanical rotational friction" of a dc-servomotor may at the outset seem to mean be semantically similar, the former being used in the Simulation tool and the latter in the dataserver, but in actual fact they are different - their relationship being formally defined by a mathematical one.

10.2 Tool Integration

Figure 10 shows the AI environment as the focal point for information movement and acts as a blackboard structure for the other tools. The figure illustrates a typical flow of information when fully specifying a generic component object in a scheme. The Selection/Matching tool queries the dataserver for information about model types of a component and after a selection is performed the record number for the specific model type is transferred across to the generic component object, which will now send a message to its associated dataserver to download all necessary attribute values.

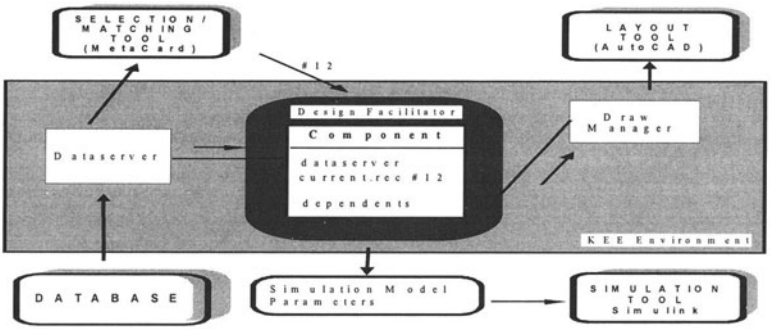

Figure 10 Schemebuilder's integrated infrastructure.

A slot called "dependants" in each generic component object stores a list of all objects that have a dependency on the generic component object. Whenever the current record slot represented as "current.rec" in Figure 10 is updated, each dependent object is sent a message to update itself. This mechanism is essential to provide a co-ordinated update of models so that data consistency between models is maintained.

11 CONCLUSIONS

Design is an integrated and integrating process during which the designer must call upon vast amounts of knowledge and experience in a systematic manner to develop the design of a product from first principles to a satisfactory prototype. It is a combination of rigorous intellectual discipline and creative thinking. In the development of the integrated environment, we have been very conscious of two important factors: (1) the need to allow the designer or designers to have total control over the design process at all times, allowing them to compare the alternatives in a clear and unambiguous way, and (2) to provide the underlying core structure of data and knowledge in such a way that the designer is not hindered in his actions and may freely develop new ideas and procedures, whilst at the same time being equally free to revisit previous decisions, whether in respect to the FEST-ER development or aspects of matching or simulation. The provision of a sophisticated audit trail of the designer's action and the advice proffered or sought is important in this respect.

The greatest importance of the work that has been described must lie in the manner in which it has been integrated to allow the complete design process to be undertaken in one homogenous computer environment. The example given throughout this paper, of the design from "first principles" of a portable metabolyte infuser for use with human subjects, has demonstrated how the process may be used not only to ensure the creation of a number of workable alternative designs but how a design, once chosen, may readily be constructed as a physical prototype from the list of chosen components, the block diagram of their interconnection, their physical layout and the spatial design of the special components and casing taken from the 3D solid model. The provision of a sophisticated Audit Trail of the designer's action and the advice proffered or sought is important in this respect.

Arguably the single most compelling reason for an engineering enterprise to make the paradigm shift to concurrent engineering, difficult and sometimes painful as it may be, is to reduce as much as possible the time-to-market factor for a product: the earlier a product can be brought into the market the greater the market share would the company potentially capture. It is often observed that the concurrent engineering approach engenders a more protracted design phase than in the traditional serial engineering approach. This can be attributed to the fact that more design iterations occur at the design phase than traditionally would be: designing it right the first time is the prime concern. To reduce lead time further it is sensible that designers be provided with greater computer support at the conceptual design stage. Furthermore, an interdisciplinary approach to design should be encouraged so that the optimal allocation of functions to different engineering domains can be achieved more successfully. The Schemebuilder project addresses this problem by providing an integrated design workbench for the designer (novice or experienced) to take an "expression of need" and developing it into schemes very quickly which can then be quantified and evaluated to assess the impact of the decisions made. This has been achieved by adopting a unifying representation via bond graphs for modelling of designs and supported by a set of integrated design tools made up of knowledge based systems, simulation packages, CAD packages and hypermedia systems.

The AI tools that have been demonstrated have concentrated on the energetic aspects of conceptual design and have built on the unique properties of Bond Graph Theory to provide robustness and rigour to interdisciplinary design. In our future studies we will be extending this thinking to information domains. Research into the nature of the information aspects of design is indicating that a similar approach to that for energetic systems will prove successful in providing the designer with

much needed help to handle the information domain, in which ever form that information is transmitted.

In presenting this paper we have sought to demonstrate the potential for AI tools of many different forms to be integrated into a homogenous whole to provide support throughout the conceptual design process. This we believe we have been able to do. It is now for us to build on these foundations and extend to a level where the designer in industry feels comfortable and confident with this support.

12 ACKNOWLEDGEMENTS

This paper represents a considerable effort over a period of years by a large number of individuals. The author would like to thank the staff of the Engineering Design Centre for all their help and enthusiasm for this project and the Engineering Design Committee of the Engineering and Physical Sciences Research Council for their continuing generous financial support.

13 REFERENCES

Andreasen, M. M., (1980). *Syntesemetoder på Systemgrundlag.* PhD, Lund Technical University, Lund, Sweden.

Bracewell, R. H., et al., (1993). Schemebuilder, A Design Aid for the Conceptual Stages of Product Design. *9th International Conference on Engineering Design, ICED '93*, The Hague, pp 1311-1318.

Bradley, D. A., et al., (1993). Engineering Design & Mechatronics - The Schemebuilder Project. *Journal of Research in Engineering Design.* 4(4): pp 241-248.

Buur, J., (1990). *A Theoretical Approach to Mechatronics Design.* PhD, T.U.Denmark, Lyngby.

Filman, R. E., (1988). Reasoning with Worlds and Truth Maintenance in a Knowledge-Based Programming Environment. *Communications of the ACM.* **31(4)**: pp 382-401.

Finger, S. and Rinderle, J. R., (1989). A Transformational Approach to Mechanical Design Using a Bond Graph Grammar. *Design Theory and Methodology - DTM '89.* DE-Vol 17. W.H. Elmaraghy. pp 107-115.

Fischer, G., (1990). Communication Requirement for Co-operative Problem Solving Systems. *Journal of Information Systems.* **15(1)**: pp 21-36.

Fishwick, P. A., (1989). Qualitative Methodology in Simulation Model Engineering. *Simulation.* **52**: pp 95-101.

French, M. J., (1985). *Conceptual Design for Engineers, 2nd Ed.* London: Design Council.

IvyTeam., (1993). *SystemSpecs 2.1 Reference Manual.* Zug, Switzerland.

Jensen, K. and Rozenberg, G., Ed. (1991). *High-Level Petri Nets, Theory and Application.* Springer-Verlag.

Karnopp, D. C., et al., (1990). *System Dynamics: A Unified Approach, 2nd ed.* Chichester: Wiley.

Oh, V., (1993). *Intelligent Design - Assistant Systems for Engineering Design.* EDC2. Lancaster University.

Paynter, H. M., (1961). *Analysis and Design of Engineering Systems.* Cambridge, USA: MIT Press.

Rychener, M. D., Ed. (1988). *Expert Systems for Engineering Design.* Boston, Academic Press.

Sharpe, J. E. E., (1978). Bond Graph Synthesis of Robots & Telechirs. *3rd CISM IFFTOMM Conference in Robotics & Manipulators*, Udine, Italy, pp 168-176.

Sharpe, J. E. E. and Bracewell, R. H., (1993). Application of Bond Graph Methodology to Concurrent Conceptual Design of Interdisciplinary Systems. *IEEE/SMC Conference*, Le Touquet.

Soderman, U. and Stromberg, J., (1991). Combining Qualitative and Quantitative Knowledge to Generate Models of Physical Systems. *12th Int Conference on Artificial Intelligence*, Sydney, pp 1158-1163.

Top, J. and Akkermans, H., (1991). Computational and physical causality. *Proc 12th Int'l Joint Conf on Artificial Intelligence*, Sydney, pp 1171-1176.

Ulrich, K. T. and Seering, W. P., (1989). Synthesis of Schematic Descriptions in Mechanical Design. *Research in Engineering Design.* **1(1)**: pp 3-18.

Yourdon, E., (1989). *Modern Structured Analysis.* Englewood Cliffs, New Jersey: Prentice-Hall.

14 BIBLIOGRAPHY

Professor John Sharpe has a special interest in System Design. He has a PhD from Cambridge University on Optimal System Design and after a period as Senior Assistant in research at Cambridge, Engineer with the National Institute for Medical Research and 16 years with the University of London, has directed the work at the Lancaster EDC since 1992. He has special interests in graph theory, particularly Bond Graphs, and first published a paper on Bond Graph Synthesis in 1978. He is a Fellow of the Institute of Mechanical Engineers and a Member of the Institute of Electrical Engineers. Currently has a serious hobby in the development of waste-to-energy techniques and strategies.