J. S. Gero and F. Sudweeks (eds), Artificial Intelligence in Design '96, 603-622.

GENERATING CONCEPTUAL SOLUTIONS ON FUNCSION: EVOLUTION OF A FUNCTIONAL SYNTHESISER

AMARESH CHAKRABARTI AND MING XI TANG
Engineering Design Centre, Department of Engineering
University of Cambridge
Trumpington Street, Cambridge CB1 1PZ, UK

Abstract. FuncSION is a software that can synthesise, using a database of functional elements, an exhaustive set of solution concepts to satisfy functional requirements of a design problem. It is intended to stimulate designers' thinking by providing a framework where these solutions are offered to the designers for exploration in the conceptual design stage. Reported in this paper are some of the testing results using FuncSION in two case studies and three hands on experiments, in terms of its ability to (i) offer a wide range of new, interesting and useful ideas, and (ii) facilitate exploration of these ideas in an effective way. The main results are: it does provide useful ideas and interesting insights to the designers, but does this at the cost of having to deal with a potentially huge list of candidate solutions which are hard to explore sufficiently. Based on these results, a scheme for coping with a large number of solutions without losing explorability is proposed, whereby designs could be generated and explored at multiple levels of abstraction, using pre-defined as well as customised clustering strategies at any of these levels. An implementation of the scheme, in terms of a user editable hierarchical database of elements and solutions, and a general algorithm for synthesis at multiple levels are proposed. A set of clustering strategies for identifying and grouping the solutions considered by experienced designers to be redundant and wasteful is also discussed, with some initial testing results.

1. Introduction

In transmission design, the major functional requirement of a design is to transmit and transform forces and motions. This can be expressed as a transformation from a set of input characteristics to a set of output characteristics. Each of these characteristics may be required to change with time. The transformation at an instant between the input and the output characteristics is an *instantaneous transformation*. An ordered set of such transformations can be used to express the overall functional requirement of a problem. In this approach, a *solution concept* is an abstract description of a

system, of identifiable *functional elements,* that can satisfy given functional requirements. For instance, a solution concept, for transmitting a force on the same plane but into a different direction and position, could be a system in which an input rack takes the input force to rotate a pinion, which moves an output rack in the required direction to provide the required output force.

The instantaneous transformation of a given system can be deduced using the information about its constituent elements, connections, and their rules of combination. FuncSION (acronym for **Func**tional **S**ynthesiser for **I**nput **O**utput **N**etworks) is a system developed at the EDC in Cambridge University that synthesises solution concepts, using functional elements from a database as illustrated in Figure 1, to fulfil a given functional requirement of a design in terms of its required instantaneous transformation, so that designs so synthesised fulfils the required function at one instant of time.

In the representation that FuncSION uses, a design problem is defined as a transformation between a set of instantaneous *input* (and *output vectors*, each of which has a set of characteristics such as *kind* (the type of I/O, such as force, rotation etc.), *orientation* (the spatial axis along which an I/O is oriented) *sense* (the sense of the I/O along the spatial axis) a *position* (spatial co-ordinates) and *magnitude,* to represent the required I/O characteristics. FuncSION allows the user-definition of a database of functional elements, where each element is expressed as a transformer which transforms an input vector of given characteristics into an output vector of specified characteristics (for example, a screw element can transform an input vector of the rotation kind into an output vector of the translation kind so that they are co-axial to each other).

In FuncSION, the synthesis of a solution concept is supported in a three step process. The first step involves Kind Synthesis, where an exhaustive search algorithm is used to synthesise a set of topological networks of causally connected functional elements, each of which is structurally feasible and can transform the give input kind into the output kind required of a design problem. Concepts generated by this procedure is exhaustive, i.e., all possible combinations of elements, from a given database of elements and using a specified maximum number of the transformations allowed in any solution concept, which fulfils the given functional requirement, are generated. In the second and third steps, possible alternative spatial configurations (i.e., spatial layouts) of each such topological candidate solution concept can be generated, using two further procedures called orientation and sense synthesis, so that the concepts can also satisfy the orientation and sense constraints imposed by the design problem. The representations for the design problem, design solution concepts and synthesis procedures used in FuncSION are reported in Chakrabarti and Bligh (1994).

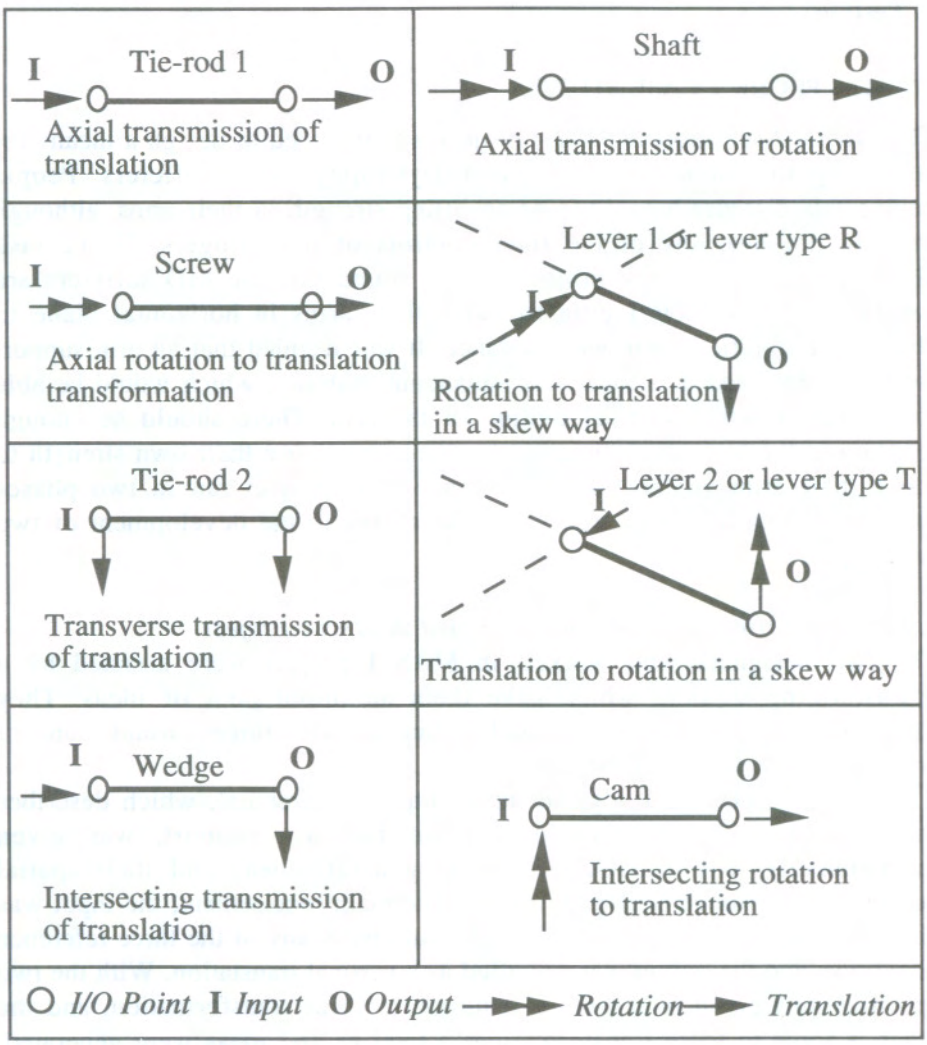


Figure 1. The elements used by FuncSION in the synthesis of conceptual solutions.

The objective of this paper is to present the development of FuncSION based on its application in real design cases and its evaluation by experienced designers. In Section 2 the results of testing FuncSION are discussed. From these tests, the main research problems to be tackled before utilising the potential of FuncSION are identified in Section 3 and Section 4. A scheme for solving these problems and its implementation is presented in Section 5, with some of initial test results.

2. Testing

2.1. MAS PROJECT CASE STUDIES

The Mobile Arm Support (MAS) project was intended to design a means for enhancing the mobility of Muscular Dystrophy (MD) sufferers. People having this disorder have little or no lifting strength in their arms, although they do not lose any of the finer controls of their fingers. In the task clarification phase of the project, it was found that the MD sufferers are capable of using their inertia to move their arms in horizontal plane in absence of significant surface resistance. It was decided that an arm support would be designed as a means of enhancing mobility, which would be able to provide powered vertical motion of the arm. There should be enough freedom in the horizontal direction for the users to use their own strength to move their arms in the horizontal plane. The project ran in two phases spanning a total of over three years, which led to the development of two prototypes.

2.1.1. Comparison with Designs generated in MAS I Project

The two designers who worked in MAS I project were assisted by a brainstorming session, which gave them an initial pool of ideas. They explored these ideas, and eventually came up with three variants, one of which was selected for embodiment.

As a retrospective study, an input-output requirement, which describes the intended instantaneous function of the arm support, was given independently to FuncSION, for it to generate ideas and their spatial configurations. As vertical mobility was the main requirement, the input was either a translation or a rotation, which could be in any of the three reference directions, and the output was specified as a vertical translation. With the two specifications given to it (one is a torque to force transformation, and the other, a force to force transformation) a total of 162 ideas were generated. These ideas were compared with those that designers generated.

There were a total of 73 ideas generated by the designers. Most of these ideas are either physical effect-like solutions, or incomplete and incomprehensible, or from a different domain of knowledge, and thus are not within the realm of FuncSION, leaving a total of 27, not necessarily distinct ideas which could be compared with the solutions FuncSION generated. FuncSION managed to generate 22 of these.

Of the 162 solutions that FuncSION suggested, 13 (the number of distinct ideas of the 22) were generated by the designers. However, given that FuncSION allows the same element to be used more than once in a solution, it often generates a number of solutions which might be considered as variants of other ideas (possibly giving an inflated impression of its

originality). Also, some solutions generated might be too expensive to be considered by the designers at all. One method to compare designers' solutions with solutions generated by FuncSION, in the above context, might be to group designs generated by FuncSION into a number of clusters and to eliminate those clusters which contain "expensive" solutions. If an idea exists in the designers' documents which can be abstracted as one of the solutions in a cluster from FuncSION, then to assume that this cluster has been considered by the designers. There are two problems with this method. One is the issues of what criteria should be used to group designs as similar, and to classify designs as wasteful / expensive. The second is that the designs generated by the designers are often at a different level of abstraction than those generated by FuncSION. If these solutions are at a higher level of abstraction, these cannot be discussed within the realm of FuncSION (e.g., those ideas that are physical-effect-like). If these are at a lower level of abstraction, we need to abstract them to the right level before these can be compared with solutions generated by FuncSION; this has two consequent difficulties.

Take the instance of the "shaft rack and pinion" solution (Bauert, 1993) as an example, this could be interpreted as a "shaft lever tie rod" solution before it is compared with FuncSION (because a pinion and a rack could be abstracted as a lever and an axial tie-rod respectively).The difficulties are that if we interpret, by having spotted an instance of the "shaft rack and pinion" design, that the designers have considered the whole cluster that represents "shaft levers tie rod", then we might have made two layers of mis-conception: whether or not the designers considered "shaft lever tie rod" solution class as a whole (various possible embodiments of this class at the level of abstraction at which the designers considered their design); whether or not they considered the whole cluster of "shaft lever tie rod" type solutions (the variants of this solution at the same level of abstraction such as "lever tie rod", "shaft lever" etc.).

We have tried to deal with the first problem, of finding criteria for clustering designs, by carrying out a set of further experiments with experienced designers, and identifying their common notions of wasteful/expensive and similar as clustering heuristics. The second problem, about comparison of designers' idea-instances with FuncSION's solution clusters, was dealt with by this assumption that if designers' instances can be abstracted into more than one solution in a cluster, then they must have considered the solution types represented by that whole cluster. However, if there is just one single or no idea-instance that could be abstracted as a solution in a cluster from FuncSION, then the designers did not consider this solution cluster.

There were interesting and inexpensive solutions that were suggested by the computer, which designers did not conceive (one example of which is a

single link lever connecting an input rotation to a tie-rod via a four bar linkage to provide an output hand motion). It was interesting to note that some solutions which were regarded by the designers as distinct solutions were regarded by the computer as topologically the same (e.g., the final two solutions in MAS I, see Figure 2). This signifies the importance of considering spatial configurations as distinct solutions.

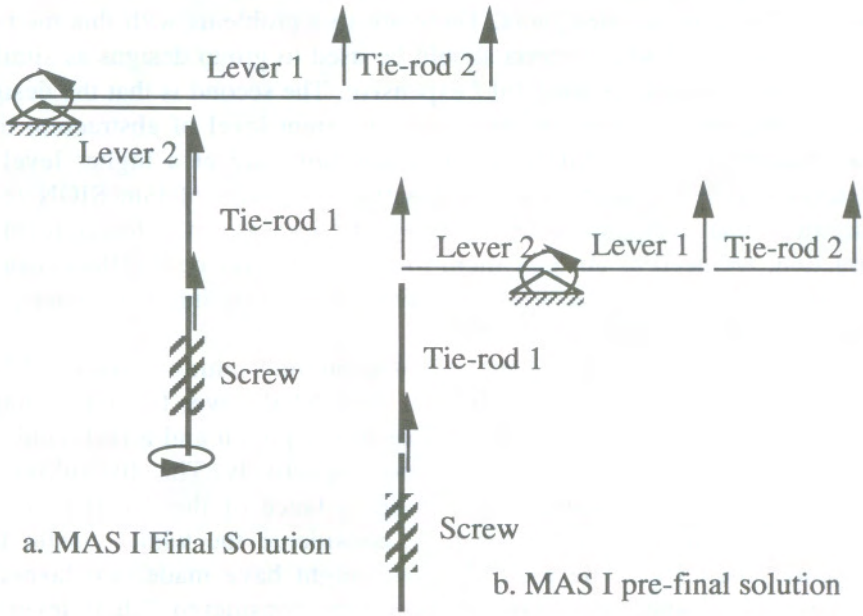


Figure 2. The final solutions in MAS I project.

2.1.2. Comparisons with Designs Generated in MAS II Project

In phase two of MAS project, designers were given the designs generated by FuncSION in Phase I, along with the other existing ones, for consideration. They went through these as an exercise, hardly taking note of them as serious solutions, and got on with designing as they otherwise would (and again did not come up with those feasible designs as in phase I). Possible reasons might have been that (i) right from the beginning this was taken as a redesign exercise, with the intention of modifying the previous designs to alleviate the existing problems; (ii) concepts generated by FuncSION were not easy to understand due to their user-non-friendly abstract representation, and lack of visualisation of how they worked; (iii) there were too many solutions to browse; (iv) there were a large number of infeasible, expensive, or similar solutions which discouraged the designers to explore further. However, these are only guesses, and needed validating before they can be

given serious consideration. We thus did some further testing for an evaluation, which is discussed below.

2.2. HANDS ON EXPERIMENTS BY EXPERIENCED DESIGNERS

Three experienced designers were asked to use the system to evaluate the solutions generated by the computer for aspects of their originality, feasibility, redundancy and wastefulness, and to comment on whether and how they would modify the ideas they find unacceptable to make them acceptable. They were also asked to make comments on the ease of use of the package, and to make any other observations or suggestions.

2.2.1. Experience of Designer A

Designer A went through the MAS design exercise twice using a different database of basic elements each time. In the first experiment he used 5 elements: two lever types, two types of transitional elements, and a screw type element. He wanted to check whether or not FuncSION produced the solution he had in mind, which it did. He then went through the second experiment, where he used another database of five elements. He could not think of any sensible solution using these elements, and wanted to check whether FuncSION could surprise him. There were three solutions which he found useful and interesting.

However, there were a large number of solutions which he thought were redundant (with repetitive transitional elements, e.g., three shafts in series as a distinct solution to two shafts) or wasteful (e.g., having two cams in a single I/O design). For instance, of the 20 solutions in this second experiment, there were 4 redundant and 3 wasteful designs, which totalled 33% of the total number of solutions.

He found that he could not cope with more than 20 solutions at a time, and suggested that browsing the solutions using user-defined categories (such as all solutions with a screw, or all solutions with lever only) would make handling large number of solutions easier. The solutions were difficult to visualise or interpret as we expected, and he thought having iconic representations coupled with simulation facilities would make visualisation easier.

2.2.2. Experience of Designer B

Designer B went through the MAS exercise thrice, each time with a further reduced database of elements and less number of elements to be used in a solution concept, so as to bring down a large number of solutions to explore (from 700 to 50 to 6). In each of these cases, he found that a solution concept having additional tie-rods or shafts were just variations on the original theme.

In the above cases, designs with a lever preceding a screw were considered wasteful, as long as levers were being interpreted as links and not as temporal elements such as gears. Once this bias was removed, however, some of the solutions having levers preceding gears were now considered geared variations of the rest of the solution. Similarly, a solution having three levers in series followed by a tie rod was originally considered as a feasible but not exciting solution when the levers were interpreted as links. But when the levers were interpreted as gears, the designer found the same solution a clever new idea as this became a rack and pinion solution. This means that being able to see the solution at levels of greater detail reveals more insights as to how useful it might be. He felt that the solutions in these exercises gave him six distinct ideas. The first one consisted of two cams, which he felt, unlike Designer A, was not a wasteful idea, but a new idea he did not think of. The other clusters were screws with tie-rods and levers, levers and tie-rods, two cams connected by a lever, a cam driving two levers, and a cam connected with tie-rods, which included both MAS I and MAS II final solutions! Having a vertical tie rod on a screw gave him the idea of using a sleeve to isolate the transitional component of the screw (an insight).

Regarding visualisation issues, levers were not understood as being capable of being abstractions of gears, in the beginning. Also, a lever, as used in FuncSION, was not a conventional see-saw type but a more fundamental element which could be combined in various ways to produce bell crank levers as well as see-saws. Cams, in the present representation, were hard to visualise, and it was hard to visualise tie rods as axial links.

Regarding the procedural bits, Designer B felt that when he found an interesting Cam based design, he wanted to explore all the Cam based designs. So a user-defined clustering facility such as find all designs that have a shaft in the middle would be useful. He suggested that it would be useful to do synthesis with only output specified.

2.2.3. Experience of Designer C

Designer C went through the MAS exercise twice. In the first run, he chose three elements from the database, and asked for solutions having at most three elements. There were two solutions: one with two levers and the other with two levers and a cam in between. He expected both the solutions and there were no surprises, although both were perfectly reasonable solutions. He then chose a 4 element-database, and solved again for the same requirements. This time there were twenty solutions, several redundant (the heuristic was that one or many translators in series, or at the beginning or end), though he thought these variations might be useful for optimisation purposes. Here also, the solutions did not surprise him. This is not surprising because the database chosen was very limited and there was not much scope

for innovation. He felt that visualisation would be improved if the symbols were more self-explanatory, and if two solutions could be seen alongside one another.

3. Observations and Discussion

Designers in the above experiments found that FuncSION in general generates a range of interesting solutions, and often comes up with surprisingly clever ideas and insight. However, it also generates a large number of redundant and expensive solutions, and this makes it difficult or frustrating to evaluate and explore the ideas to any depth. They had some difficulty in visualising the designs at the present representation, and could visualise only when these designs were shown at a lower level of abstraction.

On the whole the above experiments suggested a common pattern of when solutions were considered similar: if two solutions are different only by a transitional element (e.g., a tie rod, or a shaft), then they are similar. The consideration for wastefulness was not as straightforward, however.

If this criterion of what redundant solutions mean were applied to cluster the designs FuncSION suggested in the MAS I and II cases described before, and the wasteful designs it suggested were clustered using the wastefulness criterion, the rest of its designs could fall into 12 different clusters of solutions of an average size of about 12. Of these, only 4 clusters were considered at any length at all by the designers in MAS I and MAS II, while just a single low-level instance was found for 2 of the 8 other clusters. This indicated the potential of FuncSION for suggesting different ideas and idea types. It is important to note that the above clusters were the result of solving the MAS problem using a database of 5 elements only. If this database were increased to 7 for instance, the number of clusters, after eliminating the expensive solutions would be as high as 29, of which only 6 would then have been considered by the designers, and only 4 of these at any length!

Based on the experience gained from these case studies and experiments, it was felt that FuncSION needed further attention in three different areas:

3.1. TOO MANY SOLUTIONS

One of the problems associated with the synthesis approach adopted by FuncSION is that the system may generate so many solutions that it is difficult for the designer to even browse through them. Some of these solutions were considered by the designers to be redundant (which is a variation another which uses additional, non-essential elements), or wasteful. However, the exploration of 'redundant' solutions often might be useful if the non-redundant ones cannot provide some additional functions which are not within the realm of the main function.

Take the solution that was generated by FuncSION for Phase II of the MAS project (see Figure 3) as an example. There are two consecutive tie-rods of the same type, which might appear to be redundant unless one were trying to provide two extra degrees of freedom for the movement of the output point in the horizontal plane.

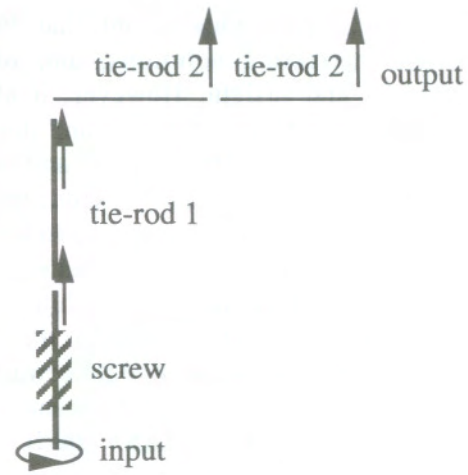


Figure 3. Redundancy can be useful.

Take the MAS I project as another example in which the final two solutions are considered by FuncSION to be topologically the same (Figure 2). The two solutions are only different in terms of the sense configuration. However, for the designers, one was considered to be a substantial modification of the another as it made the design more compact. So, whether a design is to be considered redundant or not, depends largely on other requirements that the design might have. Also, the exploration of redundant solutions might be useful if these non-redundant ones can provide some additional functions which were not originally thought of.

However, it was clear that far too many solutions were typically produced by FuncSION for the designer to meaningfully explore. For example, take a typical case of synthesis where only 32 topologically distinct solutions are generated from a database of 5 elements and a single I/O function, each of these can have at least 4 spatial configurations, each of which can have at least 3 physical concepts, giving a total of 384 solutions. The conclusion is that a strategy is needed to generate or present these solutions whereby they could be browsed through without being overwhelmed by them.

3.2. TOO DIFFICULT TO INTERPRET AND TO VISUALISE

Two main issues concerning the interpretation and the visualisation of the synthesis results were considered vital by the designers who evaluated the FuncSION system. The first is that the representation of elements in the database is too abstract. For example a shaft looks similar to a screw in terms of input/output function. The second is that the static representation for functional elements and conceptual solutions makes it hard for the designer to image the likely behaviour of an element or a conceptual solution, thus contributing little towards supporting designers' creative thinking. That is, the expected behaviour of each element or a solution needs to be visualised in order to give the designers more information. Thus a means of visualising solutions and their elements should be developed.

3.3. SOME DESIGNS DO NOT FUNCTION TEMPORALLY

So far, all the solutions that FuncSION generated work at one instant of time. For instance, a lever type element represents a transformation from an instantaneous input to an instantaneous output. This could be an abstraction of a gear or belt type element, which can provide translation at that point for an extended length of time, or it could be a link type lever whose position and direction of output change with time. The conclusion is that a temporal reasoning facility is required to evaluate the potential of each such solution to function temporally.

4. Objectives Revisited

The central objective of FuncSION is to provide an environment which would stimulate designers thinking by supporting them to explore a wide range of ideas and, if they wish, variants of these ideas, so as to increase their chances of developing new, interesting and useful designs. Two factors contribute to developing such ideas: there must be a wide range of computer generated ideas for them to explore, and these solutions must be explored and evaluated sufficiently by the designers.

In order to generate a wide range of ideas, FuncSION needs a wide variety of elements in its database, which in turn produces a large number of solutions, many of which are similar to each other. If a means could be developed to cluster these solutions into groups of similar solutions, and also weed out solutions that are considered "wasteful" by the designers for a specified requirement, then this number could be more manageable.

The evaluation on FuncSION indicated that it is easier to explore solutions if they are not too many, and if they can be visualised easily. It is easier to visualise a solution if it, and its component elements, can be seen at a

sufficient degree of detail (in terms of their behaviour as well as their spatial relations), i.e., the less abstract it is. On the other hand, the more abstract the database used by FuncSION is (i.e., where the an element can represent a large number of less abstract elements), the less the number of solutions generated will be. Therefore, there is a conflict about the right level of abstraction, see Figure 4. If it is too high, the solutions will be more difficult to visualise, and consequently to explore, whereas, if it is too low, there will be too many detailed solutions to explore. What further complicates this issue is that the "right" level of abstraction varies according to the experience of the designer. Experienced designers might only need to look at designs at a higher level of abstraction than an inexperienced designer, and still be able to imagine their details and evaluate them, while inexperienced designers might need more visualisation support, i.e., further degrees of possible detail of the solutions before they could evaluate them.

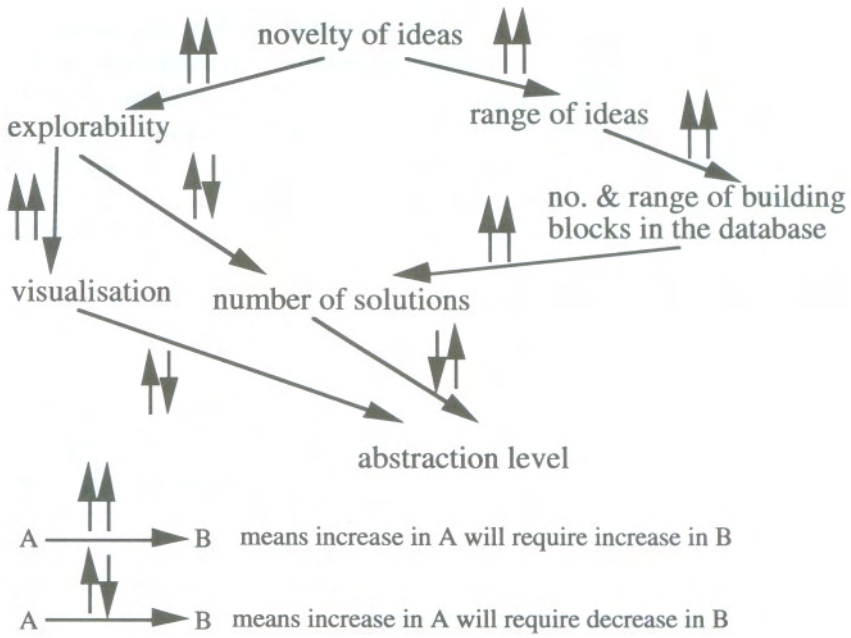


Figure 4. The factors that affect the novelty of ideas in a design.

Based on the results of evaluation, we conclude that FuncSION must be able to offer designers a wide variety of solutions to explore; the number of these solutions need to be small without compromising their range. support Designers should be visualise and browse through these designs at various degrees of detail, and should be able to see the variants a given design could have for he evaluates them. They should be able to put their own clusters on

the solution space, based on criteria such as what they consider, or is generally considered wasteful, and invoke any particular cluster at their will.

5. Further Development, Implementation and Evaluation

Four strategies have been initially identified to avoid over-generalisation of the synthesis solutions in a computer-based system. These strategies are

- to use a hierarchical functional element structure,
- to provide synthesis programs which operate at different levels of abstraction,
- to use design heuristics to cluster design solutions, and
- to provide alternative control strategies for visualisation and browsing.

A new version of FuncSION has been implemented using a knowledge-based system development tool called GoldWorksIII^{TM} on a SparcStation. It consists of a database of functional elements and their transformation rules, a functional synthesiser with synthesis algorithms that operate at different levels of abstraction, and a graphical user interface for browsing through and visualising the solution concept generated.

5.1. DATABASE

A way of eliminating unnecessary combinations of synthesis solutions is to allow the designers to choose the types of functional elements and their interfaces from a hierarchical structure. For instance, a lever element, at a lower level of abstraction, can be split into link type, gear type, pulley type, etc., while an axial tie rod can be split into axial links, chains, belts, ropes, racks etc.

Once a solution is generated at a given level of abstraction, it should be possible for the designers to navigate through the solution, or parts of it, at other levels of abstraction before making any change. The functional database should be hierarchically structured to enable the designers to edit or modify the elements at various levels of the hierarchy.

An object-oriented product data model is used to build a database of functional elements which can be selected to synthesise solutions based on a user defined input/output requirement specification. Each functional element in the database has a *type* which is associated with a set of rules that determines how it responds to different orientation or sense inputs. In the current implementation there are 72 such rules.

5.2. ALGORITHMS FOR CLUSTERING AND BROWSING SOLUTIONS

The implementation of the original version of FuncSION used specific features of the functional elements to solve the design problems at a specific

level. It is therefore unable to deal with a hierarchical structure of functional elements and therefore is abstraction-level-specific. However, it is possible to extend the algorithm so that it can solve multiple input/output synthesis problems using a set of black boxes with inputs and outputs having attributes, the exact values of which would depend on the level of the functional element hierarchy. The key idea is to separate data from the synthesis procedures, and wrap them both with a common interface. In the new version of FuncSION, a three-steps strategy is used to synthesise solutions. The first step generates an exhaustive set of candidate solution concepts for a selected set of functional elements. The second step tests the feasibility and functionality of the solution concepts to eliminate infeasible ones. The third step clusters the feasible solution concepts based on user selected heuristics.

For example, suppose we have three elements (1 2 3) (in this list each number represents a functional element and the actual elements can be filled in later). In the generate stage, all the combinations of these three elements are generated, resulting in a list of candidate solution concept structures (1, 2, 3, (1 2), (2 1), (1 3), (3 1), (2 3), (3 2), (1 2 3), (3 2 1), (1 3 2), (3 1 2), (2 1 3), (2 3 1)). In this list, (1 2), for example, means that the connection form element 1 to element 2 forms a possible solution concept structure.

In the test stage, each candidate solution concept structure is mapped to a chosen level of the functional element hierarchy in the database, retrieving the real element attributes. The compatibility of functional elements within each candidate solution concept can then be tested. This is done by removing those which are incompatible in terms of input/output transformation. For example, if the output of element 1 does not match the input of element 2, then the solution concept structure (1 2) is incompatible. All the compatible solution concepts must also be tested using the input/output requirement specification. The results of this process form the solutions of the kind synthesis step (Chakrabarti and Bligh, 1994).

Each kind synthesis solution can then be selected by the designers for orientation and sense synthesis. The orientation synthesis is done by propagating an input orientation from the input point to the output point of a kind synthesis solution concept using the orientation transformation rules, the result of orientation is a list of orientation synthesis solution concepts The sense synthesis is done by propagating an input sense from the input to the output point of an orientation synthesis solution concept. Both orientation and sense synthesis generate multiple solution concepts because one element typically responds to the same orientation and sense input in more than one way and can have alternative spatial configurations.

The outcome of kind, orientation and sense synthesis may still be a large set of solution concepts with alternative spatial configurations. The clustering

heuristics discussed above can then be selected by the designers and applied to these solutions to group solution concepts with distinct features.

A solution concept generated at one level of the functional element hierarchy can be specialised in a number of different ways. We have so far implemented the following:

1. any solution concept generated by the system at one level of the functional element hierarchy can be mapped to a lower level by substituting the elements in the solution concept with those at a lower level. This may produce a list of combinations. For example, if a solution (lever -> tie-rod) is mapped to a lower level, then for a hierarchy with two possible variants of a lever (a gear and a link-liver) and a tie-rod (a rack and a link-type-tie-rod), there will be 4 low level combinations, i.e., (gear -> rack), (gear -> link-type-tie-rod), (link-lever -> rack) and (link-lever-> link-type-tie-rod). The lower level elements may introduce interface constraints that would render some combinations invalid (in this example the second and the third solutions are invalid). A program has been designed to work out only the valid mappings.
2. a solution concept can be modified by a designer by replacing any part of it with an element or an interface at a lower level of abstraction. This allows the designers to specialise or further constrain a solution concept in a *depth-first* manner. For example, if a solution concept contains an a lever, then it is possible for the designer to modify this element by looking at its sub-class or super-class elements. Any modification made by a designer is checked by the system to ensure the consistency of the solution.
3. solution concepts generated at a low level can be clustered into a higher level by merging low level elements or interfaces into higher level ones.

It is necessary to integrate design heuristics into the synthesis process in order to offer the designers a wide range of solutions and their variants in a controlled manner. We define a *variant solution* in the following ways:

- Designer's preference, i.e., the solutions that a designer would consider as the variants of another design,
- Generalised solution concepts based on experiments, i.e., what we have found universally as variants from the hands-on experiments (this can grow as one does more experiments with the designers), or
- Variations of past design examples even though they may have been noted by the designers as wasteful. Here a wasteful solution is the one considered by the designers as inefficient or too expensive.

All these could form part of a library of heuristics or filters that could be integrated with a systematic synthesis program to weed out the solutions which may be generally regarded as being "bad ideas". This results in an organised concept solution tree instead of a huge number of solutions at the same level of abstraction.

A number of heuristics have been found useful in the experiments and can be selectively (by the designers) applied to the synthesis program to cluster the solutions generated by the computer. These heuristics include: fixing the number of transformations; each element is used more than once; each element is used no more than once; each element is used at least once; each element is used exactly once; no element is used repeatedly more than a specified times; no element is repeatedly used consecutively; same tie-rods are not directly connected; no translators such as shaft, tie-rod etc. are used; fixing the input/output elements; and only input is specified while the output is left open.

5.3. GRAPHICAL USER INTERFACE

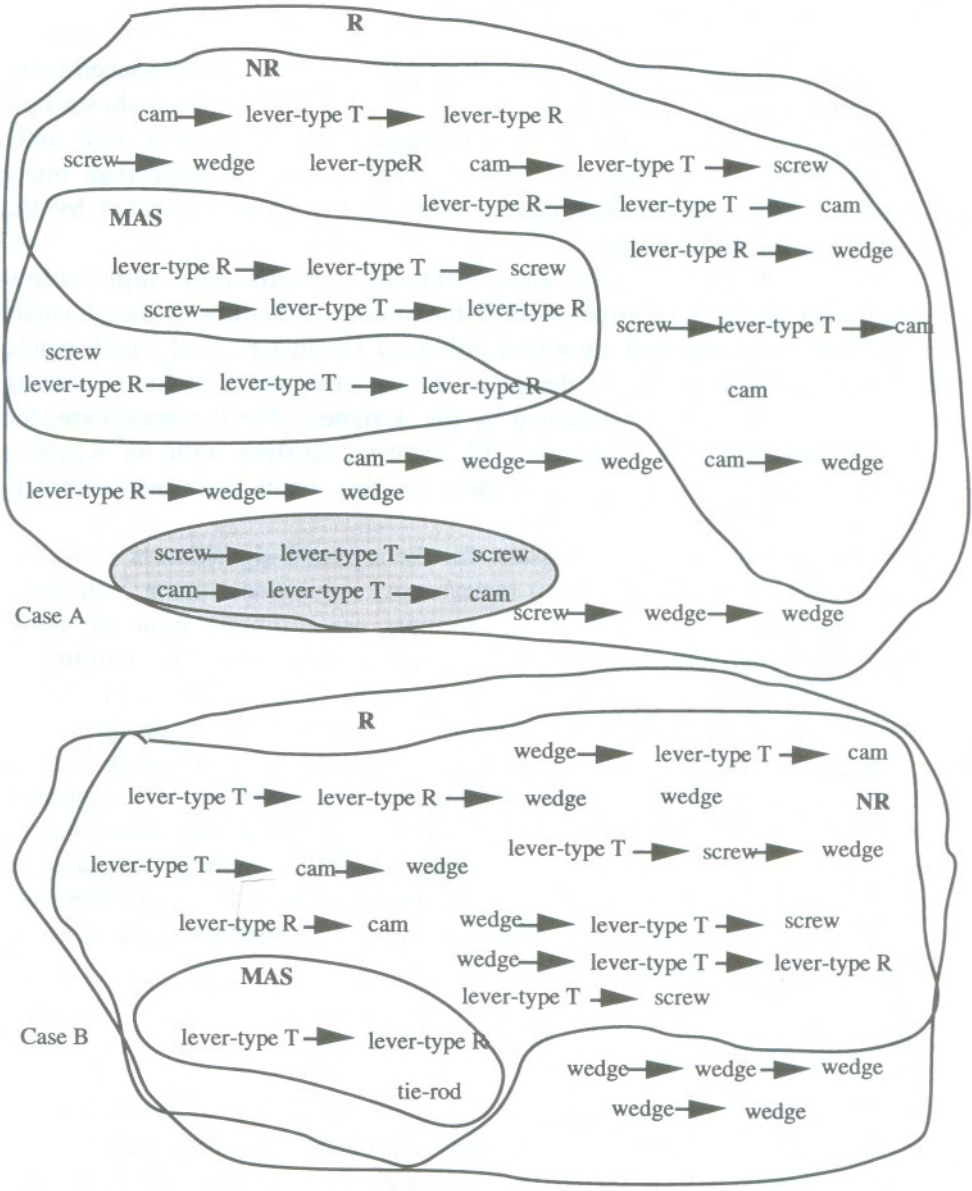
The synthesis algorithm described in Chakrabarti and Bligh (1994) used a *breadth-first* search strategy to generate general to specific synthesis solutions. While this remains a useful control strategy in the new development, a number of alternative control strategies must also be used for the designers to explore the whole solution concept tree. Some of these control strategies are:

- to allow the designers to path through all the levels of the solution concept tree;
- to pick up of a few solution concepts from a user-defined level on a random basis before generating all the possible solution concepts at that level;
- to set default values for the numbers of solution concepts to be generated at each level of the functional element hierarchy.

A new graphical user interface is designed to allow the designers to control the functional synthesis process with visualisation support by

- ionising each functional element for an easy selection;
- simulating the behaviour of individual components as well as the solution concepts generated from individual elements;
- helping the designers to browse through the solution concept tree.

Simulation is an important way of supporting the understanding of a synthesis solution so as to help with its selection and modification. While a 3D modelling tool can be used to visualise the final solution after embodiment design, it is only necessary at the functional synthesis stage to use a two dimensional graphical display scheme. Within this scheme, each functional element has an iconic image that can be actively manipulated within a graphical window.



R: idea clusters generated by the algorithm that allows repitition of elements

NR: idea clusters generated by the algorithm that does not allow repitition of elements

idea clusters that would be considered wasteful by the designers

MAS: idea clusters generated by the designers in MAS projects

Figure 5. A comparison between a Repeat and a Non-repeat algorithm.

5.4. EVALUATION OF THE NEW VERSION

In order to evaluate the newly implemented system, we have produced some test cases using some of the heuristics discussed above. Figure 5 shows how the solution clusters, produced by a repeated (each element is used more than once) and a non-repeated (each element is used no more than once) algorithm in two of the test cases, relate to the ideas generated by the designers in the MAS project.

In Case A (in the case of a rotation to translation input/output requirement using 7 elements with a maximum of 3 allowed transformers per solution), the repeated algorithm produced 18 clusters, 2 of which would have been considered by the designers as wasteful, and 5 of the remaining ones were independently generated by the designers. For the same case, the non-repeated algorithm generated 12 solution clusters with no wasteful clusters, but failed to produce 2 of the 5 clusters which were independently generated by the designers.

In Case B, the number of clusters for the repeated algorithm is 13 (in the case of a translation to translation input/output requirement using 7 elements with a maximum of 3 allowed transformers per solution), none of which would have been considered wasteful by the designers. The number of clusters produced by the non-repeat algorithm in this case is 11, which included the two that were independently touched upon by the designers.

The indication is that the non-repeated algorithm generates less number of variants or redundant solutions and thus less number of wasteful solutions, but at the cost of omitting some of the solution clusters which would still be regarded useful and important by the designers. This simply pontificates the point that it is a heuristic and not a general principle. It should therefore only be used in situations where the designers are given a prior warning about its possible consequences.

6. Related Work, Conclusions and Further Work

There are three main areas which relate to this piece of work. One is computational synthesis approaches and approaches that they take to cope with complexity, one is design methodology and how generation aspects could be supported, and the third is the systems and user interface issues.

There have been evidences in design theory and methodology that it is important to generate a range of designs and explore them sufficiently before homing in on specific designs for further development. In fact in some of the protocol studies done in the recent past, it has been found that the best approach in conceptual phase has been a consecutive expansion and narrowing down of ideas (Dylla, 1989; Fricke, 1992). It has been a major problem however, in synthesis support systems as well as in manual methods

suggested in design methodology (Pahl and Beitz, 1984) as to how to explore designs without compromising their range.

As mentioned in Lee *et al* (1992), granularity of building blocks is particularly important for managing complexity, and they felt complexity could be tackled using a few important parameters at a time. However, this is only part of the problem. Even if the problem is solved using few parameters at a time, there would still be a large number of feasible alternatives to compare, evaluate and modify. We feel that the major part of complexity arises from the conflict about level of abstraction right for getting high explorability as well as wide range of solutions. Our approach tackles this in three new ways. One is to clustering designs based on designers' heuristics of similar designs; the second is to provide range by generating solutions at a high level of abstraction, while allowing visualisation at a low level for each of these solutions, and the third is by bringing designer in the navigation process which is essential for design support systems.

In conclusion, this new version of FuncSION provides a database of hierarchical functional components and their interfaces for the user to select. The system generates synthesis solutions using an algorithm at a level of abstraction selected by the designer. The solutions generated by the system can be clustered using the heuristics chosen by the designer, allowing the designer to switch between multiple solutions and to concentrate on the interesting ones. Visualisation and simulation techniques are provided for the designer to explore and browse the hierarchical structure of functional components and the tree of synthesis solutions.

Initial testing results have shown that the integration of a hierarchical functional component database with systematic synthesis techniques, the heuristics for clustering, visualisation and simulation contributed to stimulate the designers' think. The newly developed version of FuncSION provided a good basis for utilising AI techniques in functional modelling of mechanical engineering design.

Work is being carried out to fully incorporate the control strategies and clustering heuristics discussed in this paper, and to enhance the visualisation facilities further with a fully animated graphical user interface. This new version of FuncSION is being integrated with an embodiment generator and a kinematic analysis system to form an integrated functional modelling system.

Acknowledgements

The work presented in this paper is currently being funded by the EPSRC. We would like to acknowledge the support from Dr Stuart Burgess, Dr Thomas Bligh, Mark Nowack and Doug Isgrove who acted as the designers

in the experiments reported in this paper. We would like also to acknowledge the support from Dr Nigel Ball, Dr Lucienne Blessing and Dr Tim Murdoch for their support in the development of the past and current version of FuncSION.

References

Ball, N. R. and Bauert, F.: 1992, The integrated design framework: Supporting the design process using a blackboard system, *in* J. S. Gero (ed.), *Artificial Intelligence in Design '92*, Kluwer, Dordrecht, pp. 21-38.

Bauert, F.: 1993, The mobile arm support phase in design, manufacture, testing, software tools, *Technical Report CUED/C-EDC/TR 13*, Cambridge University.

Chakrabarti, A. and Bligh, T. P.: 1994, A two-step approach to conceptual design of mechanical device, *in* J. S. Gero and F. Sudweeks (eds), *Artificial Intelligence in Design '94*, Kluwer, Dordrecht, pp. 21-38.

Ehrlenspiel, K. and Dylla, N. D.: 1989, Experimental Investigation of the design process, *in*: V. Hubka (ed.), *Proceeding of ICED89, International Conference on Engineering Design*, Mechanical Engineering Publication, Bury St Edmunds, Vol. 1, pp. 77-95.

Fricke, G.: 1992, Experimental investigation of individual processes in engineering design, *in* N. Cross, K. Doorst and N. Roozenburg (eds), *Research in Design Thinking*, Delft University Press, Delft, pp.105-109.

Johnson, A. L et al: 1993, Modelling functionality in CAD: Implications for product representation, *Proceedings of the 9th International Conference on Engineering Design.*

Lee, C-L., Iyenger, G. and Kota, S.: 1992, Automated configuration design of hydraulic systems, *in* J. S. Gero (ed.), *Artificial Intelligence in Design '92*, Kluwer, Dordrecht, pp. 61-82.

Pahl, G. and Beitz, W.: 1984, *Engineering Design*, Design Council, London.

Thornton, A.: 1993, *Constraint Specification and Satisfaction in Embodiment Design*, PhD Thesis, University of Cambridge, Department of Engineering.