# SYNTHESIS OF CONCEPTUAL DESIGNS FOR SENSORS USING SAPPHIRE-LITE

**Sarkar, Biplab; Chakrabarti, Amaresh; Ananthasuresh, G.K**
Indian Institute of Science, India

## Abstract

The demand for variety of products and the shorter time to market is encouraging designers to adopt computer aided concept generation techniques. One such technique is being explored here. The present work makes an attempt towards synthesis of concepts for sensors using physical laws and effects as building blocks.A database of building blocks based upon the SAPPhIRE-lite model of causality is maintained. It uses composition to explore the solution space. The algorithm has been implemented in a web based tool. The tool generates two types of sensor designs: direct sensing designs and feedback sensing designs. According to the literature, synthesis using building blocks often lead to vague solutions principles. The current work tries to avoid uninteresting solutions by using some heuristics. A particularly novel outcome of the work described here is the generation of feedback based solutions, something not generated automatically before. A number of patent violations were observed with the set of generated concepts; thus emphasizing some amount of novelty in the designs.

**Keywords**: Conceptual design, Computational design synthesis, Functional modelling, SAPPhIRE-lite, Feedback Sensor

**Contact**:
Biplab Sarkar
Indian Institute of Science
Centre for Product Design and Manufacturing
India
biplab@cpdm.iisc.ernet.in

Please cite this paper as:
Surnames, Initials: *Title of paper*. In: Proceedings of the 20th International Conference on Engineering Design (ICED15), Vol. nn: Title of Volume, Milan, Italy, 27.-30.07.2015

## 1 INTRODUCTION

In the current competitive market, there is a huge demand for variety in products and solutions offered by the industry, that too, within a short time to market. It has been observed that the number of concepts explored has a positive influence on the variety of solutions produced (Srinivasan and Chakrabarti, 2009). According to Pahl and Beitz, (1996), conceptual design is the phase in the process of designing, when solution principles (concepts) are developed to meet desired functions. Researchers have proposed various ways of concept generation. Computer-based conceptual design synthesis is an area of research that develops approaches for supporting fast exploration of a solution space. An overview of the computer-based design synthesis research is available in Chakrabarti et al. (2011). One such approach explored is conceptual design synthesis using physical laws and effects as building blocks (Chakrabarti and Bligh, 1996; Zavbi and Duhovnik, 2000; Srinivasan and Chakrabarti, 2009). The SAPPhIRE-lite model as proposed by Sarkar et al. (2015) is a variant of the SAPPhIRE model (Chakrabarti et al., 2005) that uses physical laws and effects, and is the focus of this paper. It has been shown in earlier literature that SAPPhIRE model of causality can be used for analysis and synthesis of conceptual designs (Srinivasan and Chakrabarti, 2009). Further, it was shown in Chakrabarti et al., (2013) that SAPPhIRE model can be used to capture different views of function within a generic model of designing. Function manifests at the various outcome levels of abstraction of the SAPPhIRE model.

Sensors are a class of devices whose function manifests primarily at the effect level of the SAPPhIRE model. The paper focuses on automated synthesis of a variety of concepts for sensors using the SAPPhIRE-lite model of causality. The aim is to offer this to designers of sensors to explore a large solution space, thereby increasing their chances of developing solutions of greater novelty and value.

## 2 RELATED WORK

According to the existing literature, there are two broad approaches in computer aided conceptual designs; on the one extreme is the case-based approach (Prabhakar and Goel, 1998) and on the other extreme is the synthesis-based approach. In the case-based approach, modification of existing solutions to meet new functionalities is practised. This often leads to biasing towards familiar solution principles (Schmitt, 1993) and therefore lacks variety. The synthesis-based approach falls into two categories: function-based synthesis and grammar-based synthesis. The function-based synthesis approach first proposes a functional model, and then uses this model to develop solutions. In grammar-based synthesis, the focus is to develop a formal grammar which has a set of rules that acts on a design vocabulary and transforms an initial design into a variety of new designs.

A number of researchers (Chakrabarti and Bligh, 1996; Zavbi and Duhovnik, 2000; Chakrabarti, 2001; Rihtarsic et al., 2012) have used physical laws and effects as basic building blocks in order to develop functional models. This often generates solution principles with a greater variety and uniqueness; but occasionally, one might end up with a number of vague solution principles (Chen et al., 2013). According to Chen et al., (2013) the reason for this is improper representation of physical quantities; they have achieved some progress in this direction by associating a flexible set of attributes to physical quantities. Physical laws and effects have been used in TRIZ methodology (Yan, Zanni-Merk, Cavallucci and Collet, 2014) in the form of a catalogue, to support innovative thinking, but they are not used to automatically generate concepts. Campbell (2000) studied synthesis of electro-mechanical devices using agents and evaluated them quantitatively using a catalogue of standard components. Graph grammars have been used by researchers for synthesis too (Kurtoglu et al., 2010; Helms et al. 2009; Kurtoglu and Campbell, 2006; Starling and Shea, 2005). The current work attempts synthesis of concepts for sensors using physical laws and effects as building blocks. It uses composition to explore the solution space and tries to avoid uninteresting solutions using heuristics. A particularly novel outcome of the work described here is the generation of feedback-based solutions, something not generated automatically before.

## 3 SAPPHIRE-LITE

SAPPhIRE model of causality (Chakrabarti et al., 2005) has been used successfully by various researchers for analysis and synthesis of concepts. It has also been able to incorporate various views of function within its entities. However, SAPPhIRE model was never used before to describe or

synthesise designs with feedback and or to support quantitative analysis, where the magnitude of the change produced by a solution principle could be estimated as part of synthesis. In order to support these, SAPPhIRE-lite was introduced by Sarkar et al., (2015). Similar work was done by Campbell, (2000) for electro-mechanical devices where synthesis of simple designs with quantitative evaluation was demonstrated; but no feedback designs were reported. The SAPPhIRE-lite model introduces three important changes: it uses an adaptation of SAPPhIRE, to focus on areas of interest in sensor synthesis; it concatenates multiple SAPPhIRE instances to represent changes that take place in complex scenarios, and uses measurable quantities that undergo change in these scenarios. The model is introduced below.

SAPPhIRE-lite specifically focuses on the input, organ and effect layers of the SAPPhIRE model and generalizes them to describe complex scenarios by which sensors work, including those involving feedback. It has changed the nomenclature and scope of some of the entities of the SAPPhIRE model to support quantification and to capture such scenarios.There is a strong similarity between the effect level of the SAPPhIRE model and an input-output network as in e.g. Control System's Network Theory (Kuo and Golnaraghi, 2003). Thus concepts from Network Theory were used in the effect level of SAPPhIRE model and the variant, SAPPhIRE-lite, was created. It uses a variant of Signal Flow Graph called *Switching Flow Graph (SFG)* (Smedley and Cuk, 1994) to quantitatively relate input quantities to output quantities, depending upon favourable input conditions. SFG is an input-output relation that is controlled by logic; only when the desired logical conditions are met, the output is produced as a function of the inputs. With this change, a computer algorithm can understand the scenario when a relation is valid. It also uses *Kirchhoff's current law (KCL)* (Kuo and Golnaraghi, 2003) for algebraic addition of similar quantities, thus allowing feedback to be incorporated. The model does not incorporate *Kirchhoff's Voltage law*; thus it uses a *weaker* form of Signal Flow Graph algebra; hence it can be applied to graphs without considering flow and effort variables. Signal Flow Graph is used to capture relationships among attributes without considering as to how the relationships are *achieved or maintained*. According to literature, many researchers have used signal flow for designing in the past. In that context, SAPPhIRE-lite uses a subset of the signal flow grammar used by Nagel et al. (2007).

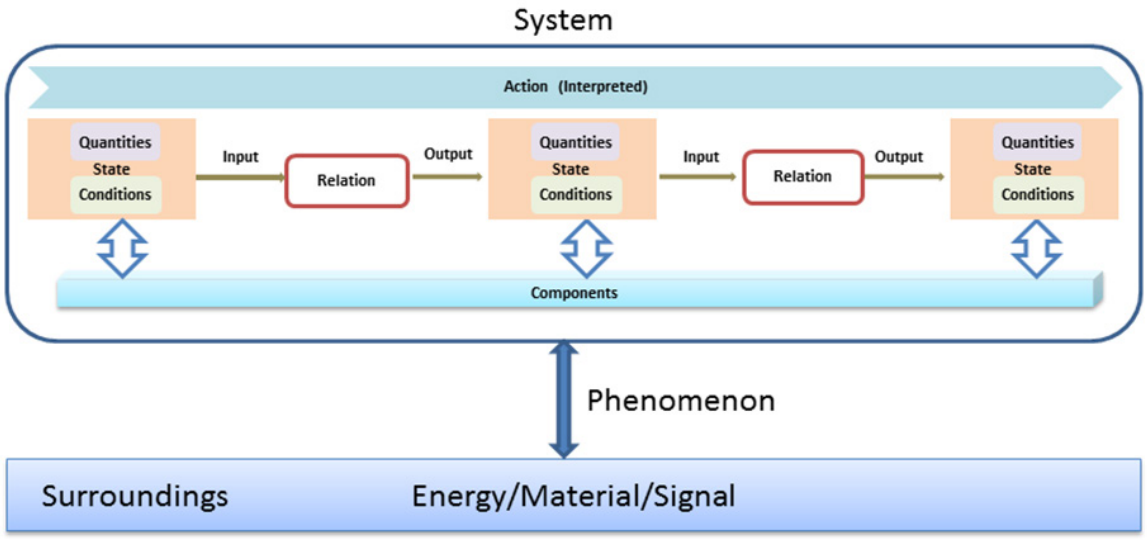


*Figure 1. SAPPhIRE-lite model instances are used to capture the working of a system*

The SAPPhIRE-lite model is described in (Sarkar, Chakrabarti and Ananthasuresh, 2015). For continuity, it will be summarized next. The model is pictorially described in [Figure 1]. It has multiple levels of abstraction. It captures the state of the system over a span of time. The state of the system is described by a set of measurable quantities and conditions belonging to the underlying level of components. Physical phenomena are the underlying exchanges of material, energy or signal between the system and its surroundings that take place due to various physical effects being activated. The exchanges in turn lead to state-changes. These state-changes are interpreted as actions. Physical effects

get activated when favourable input conditions are met and required inputs are available. We can describe a system's causal behaviour using this model as a pictorial graph. We can also carry out mathematical analysis of its state variables. Various elements of the model are summarized next.
*Quantity:* Material properties and physical attributes that are measurable are represented as quantities.
*Condition:* Under favourable conditions, a physical effect takes place when its desired inputs are met. A condition is expressed as a set of logical predicates.
*Relation*: Quantities and conditions together describe the state of a system. Physical effect changes the state of the system. A physical effect has an associated conceptual structure (Chakrabarti A., 2004; see also Rihtarsic et al., 2012). Quantities belong to that of the conceptual structure. Conceptual structures are used as a layer of abstraction of the underlying physical components. SAPPhIRE-lite refers to physical effects of the SAPPhIRE model as relations; thus generalizing it to capture any type of functional relationships across domains. A relation establishes a mapping between the output quantities to the input quantities. Thus each output quantity can be mathematically related to the set of input quantities. This allows SFG to be used for modelling relations.
*Component:* SAPPhIRE-lite refers to the parts level of SAPPhIRE model as components.
*Action:* Action is an interpretation of the state change exhibited by the system.
*Phenomenon:* A phenomenon is an abstraction of the exchange of material, energy and signal between a system and its surroundings; e.g. heat exchange. It is a result of physical effects taking place. Since SAPPhIRE-lite is based on signal flow and does not *explicitly* capture the flow of material or energy, it indirectly captures phenomena. The exchange of material and energy is captured in the form of inputs or outputs.

# 4 SYNTHESIS APPROACH

A database of building blocks needs to be maintained to synthesise solutions. One can generate solutions by chaining multiple building blocks.

## 4.1 Database

For synthesis of sensors, a database of SAPPhIRE-lite instances is maintained. Information from all the levels of the SAPPhIRE-lite model is not required for synthesis of sensor concepts at the effect level; only quantities, conditions and relations are sufficient. SAPPhIRE-lite instances of known physical effects are populated into the database. Each physical effect has a specific conceptual structure associated with it. Information about the structure is added to the database in the form of conditions. Physical effects (relations), which are a function of time only, are also incorporated into the database. Thus the database has lumped parameter models of relations which can be expressed using ordinary differential equations. In general, any functional relation can be a valid entry. So one can add abstract model of complex systems to start with and later on refine them to their details. The relations in the database are classified according to the minimum number of components required for their representation. Some relations need only one component, while others need multiple components. Thus there are two groups of relations in the database.
A system can be decomposed into a hierarchical tree of components. The component tree has three types of nodes: root, leaf and stem. Each node type exhibits a different viewpoint. The view from a leaf node is just the component and its surroundings; see [Figure 2(a)]. Similarly, a stem node has a set of components, an interconnection layer and surroundings; see [Figure 2(b)]. From the root node's view point, the system has a set of components and an interconnection layer; there is no surrounding layer; see [Figure 2(c)]. Keeping these views in mind, the relations in the database obey the following rules: (a) all outputs are in the same entity where the relation resides, (b) only input quantities can come from across entity boundaries, (c) no inputs can come from peer component entities. These rules are intended to provide a simple hierarchical view of the models created by the relations, while keeping them aligned with the reality. All relations that need more than one component reside in the non-leaf nodes of the component tree. Similarly, relations that need only one component can reside on all the nodes of the component tree. The database also stores the equations associated with the output quantities. In general, each relation has a cost attribute with unit value associated with it. To represent inheritance hierarchy among quantities, zero cost relation is used. Also quantities might have multiple synonyms, e.g. velocity is same as time derivative of displacement; for such quantity a relation with a cost of zero is used.

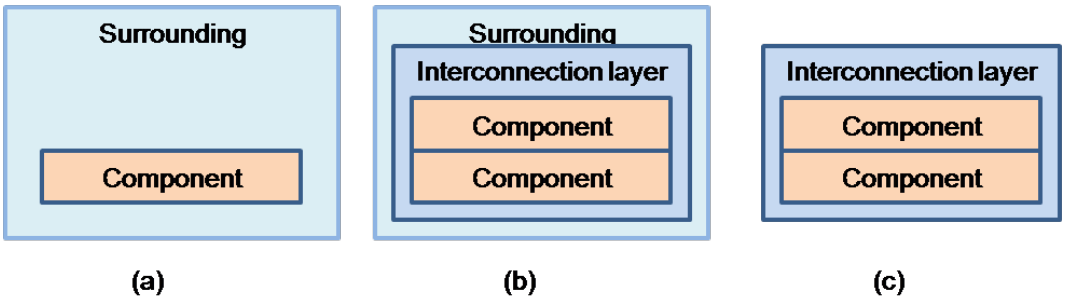


*Figure 2. View-points*

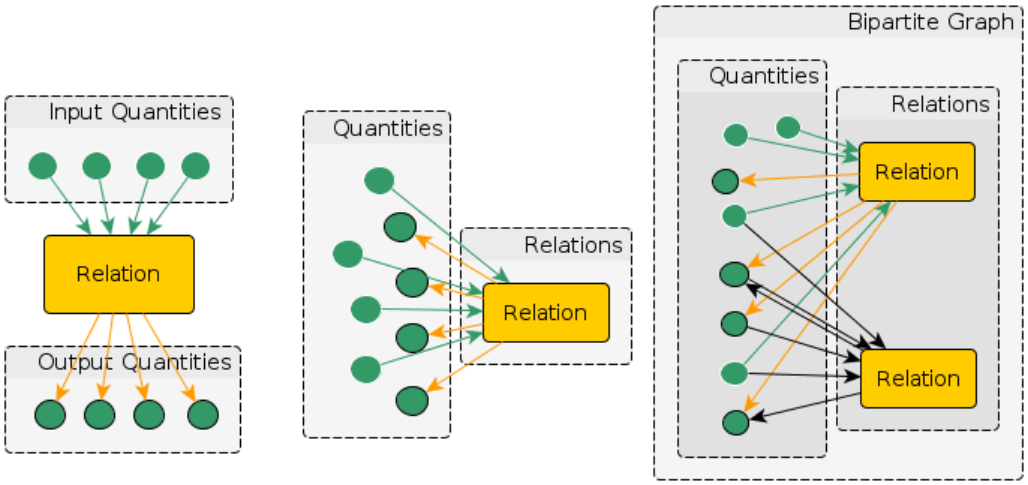


*Figure 3. Relations map input quantities to output quantities. Quantities are only connected to relations. This forms a Bi-partite graph in the run-time Database.*

The database has two versions: the *persistent* database and the *run-time* database. The *persistent* database is compact and is ideal for storage and transportation. It is used as a template to build the *run-time* database. The *run-time* database instantiates a system, called *min-system*, with minimum number of components, and fills them up with all the relations, quantities and conditions from the persistent database. The *min-system* is a system in a surrounding, where the system consists of an interconnection layer with at least one component. The synthesis algorithms use the *run-time* database. As shown in [Figure 3], the quantities and relations form a *bi-partite* graph in the *run-time* database, if edges among them are considered. In fact, exploring of concepts by using either composition or graph grammar is same as that of finding paths in the *bi-partite* graph while meeting certain guidelines. The quantities in the database are currently from multiple domains: mechanical (solid and fluid), electrical, magnetic, thermal, optical, chemical, radiation, acoustic, nuclear and general.

## 4.2 Synthesis Algorithm

The *run-time* database is used to synthesize new sensors. There are two types of sensor designs. In one case the user needs to specify both the input and the output quantity; such designs are referred as *direct sensing* designs. In the other case, the user just specifies the input quantity and feedback is used to nullify the change caused by the input quantity that is being sensed; they are called *feedback sensing* designs. The algorithm uses the *bi-partite* graph formed in the *min-system* to generate solutions.

### *4.2.1 Direct Sensing designs*

In direct sensing designs, the user specifies the quantity that needs to be sensed and also the quantity that should be the output. In the *min-system,* the quantity may exist in multiple entities; it might reside both in the interconnection layer and in a component. The algorithm looks into the *min-system* to seek

all possible combinations of the input quantity to the output quantity across all entities. This is similar to the problem of finding all possible routes between two places on a map. For this the best known algorithm is Eppstein's *k-shortest path algorithm* (Eppstein, 1998). Eppstein's algorithm is for generic directed graphs; our specific requirement of avoiding loops in the solution path adds a new constraint; thus it slows down the algorithm. Our algorithm has to run for more iterations as it filters out generated solutions based on the added constraint. The search algorithm generates a set of paths connecting the input quantity to the output quantity through a series of relations and quantities. Since we have a bi-partite graph, each path will have alternating nodes of quantities and relations. It produces a set of *bare solution principles* ranked with respect to the path cost. It is termed as *bare solution principle* because the associated *feeding graphs* are not yet explored. The *feeding graphs* are required to produce the desired inputs to activate all the relations in the *bare solution principle.* The algorithm is summarized next; for a detailed understanding of the Eppstein's *k-shortest path algorithm* see (Eppstein, 1998).

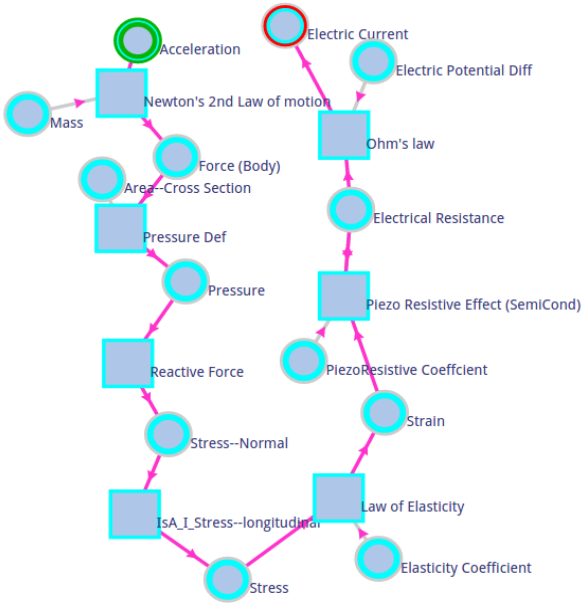


*Figure 4. Synthesized Direct Sensing design of an accelerometer that violates patent US2963911*

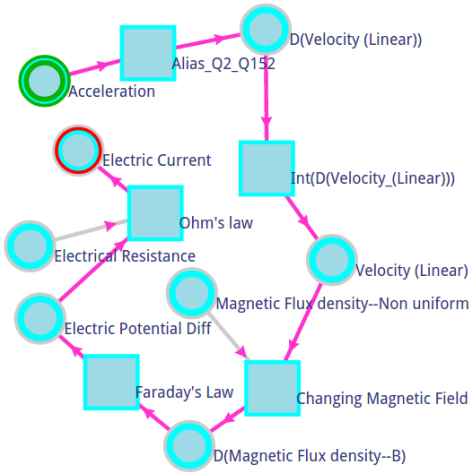


*Figure 5. Synthesised Direct sensing design of an accelerometer that violates patent US20140157897*

Eppstein's *k-shortest path algorithm* finds 'k' paths with increasing path cost between a source and a destination node in a directed graph. The algorithm has two phases. In the first phase, Dijkstra's *shortest path algorithm* (Dijkstra, 1959) is run from the destination node in the reverse direction. This causes all nodes, which have a path to the destination node, to be aware of the shortest path from that node to the destination node. All nodes which have no path to the destination node can be removed from the graph. This also assigns a shortest path for the source to the destination node. The second phase starts from the source node and tries to find all paths which deviates from the shorted path. It uses a priority queue to keep track of the paths with increasing path cost. This phase runs multiple iterations and each run produces the next best shortest path. In the presence of the added constraint of avoiding loops, some of the iterations will not produce acceptable solutions. Thus in order to find 'k' shortest paths the number of iterations might be more than 'k'. Eppstein's algorithm has a complexity of O(e + vlog(v) + k) by using a Fibonacci heap based priority queue; our algorithm has a complexity of O((e + v + k + d)log(v)) by using a binary min-heap based priority queue; here *e* represents edges, *v* represents vertices, *k* represents k-paths, *d* represents discarded solutions.

In [Figure 4] a solution principle of an accelerometer synthesised by the web based tool (Sarkar, (2014), see details in Section 5) that we have developed uses this algorithm is shown. This accelerometer works as follows. Acceleration causes the proof mass to exert force on a piezo-resistive material; the force causes pressure; pressure causes stress; stress causes strain; strain causes change in electrical resistivity of the material; and an external electrical potential is used to sense the change in

resistance in terms of electrical current. In another accelerometer principle synthesised by the tool [Figure 5], acceleration changes the velocity of the body; an induced electrical potential is generated due to Faraday's law of induction on the moving body as it has a coil in it and is placed in a spatially non-uniform magnetic field. Both sensors violate existing patents, which means the synthesis tool creates designs that are useful. In case of this example the tool produced 37 solutions principles out of which 14 patent violations were observed.

#### *4.2.2 Feedback sensing designs*

In feedback sensing designs, the user specifies the input quantity, and the algorithm searches through the *min-system* for all paths that contain a feedback loop. It should be a negative feedback to make this solution practically useful. The quantity that is fed back is nullified and one measures the input quantity in terms of the quantities in the feedback loop. These types of sensors are active in the sense that certain actuation is to be given to nullify the quantity.

To achieve all possible solutions with feedback, *depth-first search* is to be used. However, this is not always a practical solution if the database is moderately big, or if the number of solution is to be restricted to a few manageable ones and with minimal lengths. What is needed is a form of *depth-first search* that helps in discovering feedbacks but also explores all alternatives and keeps the depth of the search to the minimum. Since there is no good algorithm for this, we developed a heuristic for *depth-first search with fan-out restricted scores*. The algorithm starts with a large score and divides that score according to the number of fan-outs at each node, and the depth-first algorithm stops if the score reaches zero. Here integer scores are used. With this heuristic, a variety of solutions with minimized lengths can be synthesised. Since negative feedback is required to achieve the desired behaviour, one has to do further qualitative or quantitative analysis on the equations of the generated solution principles to ascertain the sign of the feedback. In fact, the algorithm produces a variety of *potential* solution principles with feedback. In [Figure 6], the pseudo code for the *depth-first search with fan-out restricted scores* algorithm is detailed. The algorithm uses a stack (nStk) data structure to keep track of its search path. It uses recursion to explore the graph. Node colouring is used to avoid cycles. The recursion depth is controlled by the score that is passed into it. It uses integer arithmetic, so recursion stops as soon as the score reaches zero. The score at a node is divided equally among its out-bound search paths.

```
// dfs with fan out restricted scores
nodes : Array of nodes;
nStk : stack;
id: node id;
score: interger;
function dfs(id, score) {
    if (score == 0) return;
    nStk.push(id);
    if (nodes[id].colored == false) {
        nodes[id].colored = true;
        var numOut = number of nodes[id].outEdge;
        foreach (edge E belonging to nodes[id].outEdge) {
            var dst = destination node of edge E;
            dfs(dst, floor(score/numOut));
        }
        nodes[id].colored = false;
    }
    nStk.pop();
}
```

*Figure 6. Pseudo-code for DFS with fan-out restricted scores*

In [Figure 7] conceptual design of an accelerometer synthesised using this algorithm is shown. Acceleration is sensed by a mass that converts it to body force; the force is nullified by using feedback logic; the feedback path consists of a force sensing spring that undergoes some displacement; this triggers Lorentz's effect to take place as the body is charged and is located in a magnetic field; the Lorentz's effect produces a force that acts as a feedback. Here the input acceleration can be measured in terms of the quantities that are associated with the relations in the feedback loop. [Figure 8] shows another example of an accelerometer design that uses feedback. Here the body experiences a changing

magnetic field because of the motion caused by acceleration; this causes Faraday's Law of induction to get activated in the coil surrounding the body and it produces an electric potential; this causes electric current to flow in the coil because of Ohm's law; the current caring conductor produces a magnetic field according to Ampere's law to nullify the change in magnetic field that triggered the process. This concept violates existing patent. The synthesis tool produced 35 concepts for accelerometer. A quick search revealed 2 patent violations.

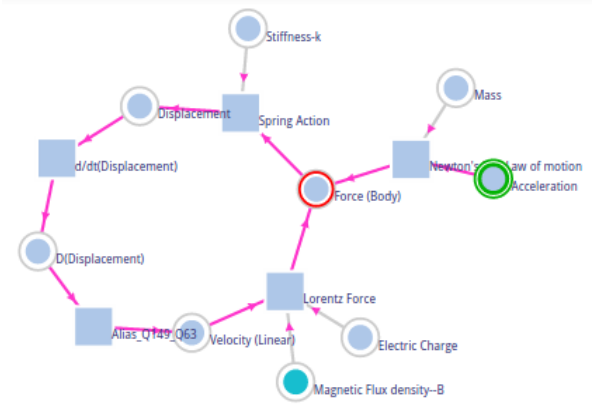


*Figure 7. Synthesised Feedback sensing design of an accelerometer*

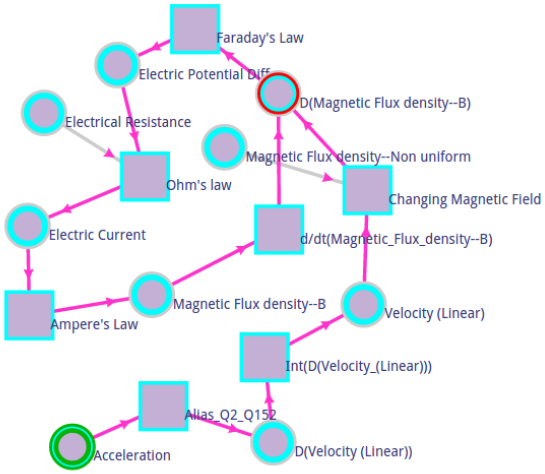


*Figure 8. Synthesised Feedback sensing design of an accelerometer that violates patent US6575029*

## 5 SOFTWARE TOOL

A web based interactive software tool (Sarkar, 2014) has been developed using the algorithms. It has a pictorial representation of the graph that models a sensor along with its quantities and relations. The database currently has 169 relations and 127 quantities. The tool uses these technologies: HTML5, CSS, SVG, Javascript libraries (AngularJS, d3, jQuery), Ruby on Rails, web workers and web sockets. It has a server that stores the database of building blocks. It uses a file-based version-controlled database (git). Since the database is shared, and multiple users can contribute to its growth, all changes to the database are traceable, and can be reverted to in case of any issues. The client side of the tool that runs on web browser does most of the work. The server only manages the database. The client's search algorithms run on the entire database. So the server sends the entire database whenever requested for. The client expands it into a run-time database and runs the search algorithms on the run-time database on a separate thread using web-workers. Each client can add, delete or modify the database and send the modified copy of the database to the server for storage. The server notifies all active clients about the modification using web-sockets. Upon seeing the database change notification, the user may request the client to synchronize the database with that of the server. The tool also provides a chatting service among all active users so as to help them build the database as a community.

The search results are presented to the user after grouping them under three categories: domain view, relation view and conceptual structure view. Each view divides the complete set of solutions into different disjoint sub-sets. For sensors, domain view is of great importance. The solutions might span across multiple domains and classifying them based on the number of domains makes it easy for the designer for proper decision making.

### 5.1 Solution filtering

The main challenge in the synthesis approach is to limit the number of uninteresting solution principles that it generates. To achieve this, the good and promising ones have to be filtered from the rest. So some heuristics are used in the tool to filter out uninteresting solutions. The tool currently uses two filters: (a) Duplicate Relations avoidance filter and (b) Duplicate Quantity avoidance filter.

Repetition of relations or quantity in the solution principle makes the solution uninteresting. Also by filtering out solutions principles with duplicate quantities, long solution principles which might have unwanted loops can be avoided.

### 5.2 Database manipulation

The tool supports addition, modification, deletion and display of the database entries. The database has some assistance to automatically generate simple relations. Often some relations are valid for the different combinations if the inputs and outputs quantities. So such relations can be automatically generated by rotating its input and output quantities. Also for quantities which are a time derivative of one another, automatic relations can be created. For quantities that might have synonyms it can generate zero-cost aliasing relations.

## 6 DISCUSSION

There are a few limitations in this approach. Capturing existing relations into the building blocks is often a challenging task. Improperly captured relations give rise to solution principles that are difficult to visualize; so modifications are often required. Refinement of understanding of various concepts is often required and the system supports it. A better understanding of relations is required to address this issue; our future work would try to focus in this direction. The generated feedback-based designs need further processing to ascertain if the solutions are practicable.

## 7 SUMMARY

The paper presents an overview of algorithms to produce concepts for two types of sensors designs, direct and feedback sensors. The approach to find out the solutions-space ahead of the solutions is also unique. Synthesis of feedback sensors is a novel outcome of this work. The algorithm uses a database that is based on SAPPhIRE-lite model of causality. It has introduced a heuristic depth-first search algorithm based on a score which explore breadth-wise keeping a shallow depth. It gives some insight into a software tool that uses the algorithms and synthesizes solution principles for sensors. It has also given some examples of generated sensor designs that have violated existing patents. There are also a set of solutions for which no existing patents were found. The approach can be applied for the synthesis of concepts for systems that involves effect level functions; one can use this approach for actuators too.

## REFERENCES


Campbell, M.I. (2000) The A-Design invention machine: a means of automating and investigating conceptual design, Carnegie Mellon University.

Chakrabarti, A. (2004) 'A new approach to structure sharing', Journal of Computing and Information Science in Engineering, vol. 4, no. 1, pp. 11-19.

Chakrabarti, A. and Bligh, T.P. (1996) 'An approach to functional synthesis of mechanical design concepts: theory, applications, and emerging research issues', AI EDAM, vol. 10, no. 4, pp. 313-331.

Chakrabarti, A. and Regno, R. (2001) 'A new approach to structure sharing', Proc. Intl. Conf. on Eng. Design, 155-162.

Chakrabarti, A., Sarkar, P., Leelavathamma, B. and Nataraju, B. (2005) 'A functional representation for aiding biomimetic and artificial inspiration of new ideas', Artificial Intelligence for Engineering Design, Analysis and Manufacturing, vol. 19, no. 2, pp. 113-132.

Chakrabarti, A., Shea, K., Stone, R., Cagan, J., Campbell, M., Hernandez, N.V. and Wood, K.L. (2011) 'Computer-based design synthesis research: an overview', Journal of Computing and Information Science in Engineering, vol. 11, no. 2, p. 021003.

Chakrabarti, A., Srinivasan, V., Ranjan, B. and Lindemann, U. (2013) 'A case for multiple views of function in design based on a common definition', Artificial Intelligence for Engineering Design, Analysis and Manufacturing, vol. 27, no. 03, pp. 271-279.

Chen, Y., Liu, Z., Huang, J. and Zhang, Z. (2013) 'A multi-agent based framework for multi-disciplinary conceptual design synthesis', DS 75-6: Proceedings of the 19th International Conference on Engineering Design (ICED13), Design for Harmonies, Vol. 6: Design Information and Knowledge, Seoul, Korea, 19-22.08. 2013.

Dijkstra, E.W. (1959) 'A note on two problems in connexion with graphs', Numerische Mathematik, vol. 1, no. 1, pp. 269-271, Available: http://dx.doi.org/10.1007/BF01386390.

Eppstein, D. (1998) 'Finding the k shortest paths', SIAM Journal on computing, vol. 28, no. 2, pp. 652-673.
Helms, B., Shea, K. and Hoisl, F. (2009) 'A framework for computational design synthesis based on graph-grammars and function-behavior-structure', ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 841-851.
Kuo, B.C. and Golnaraghi, M.F. (2003) Automatic control systems, John Wiley & Sons New York.
Kurtoglu, T., Campbell, M. and others (2006) 'A graph grammar based framework for automated concept generation', DS 36: Proceedings DESIGN 2006, the 9th International Design Conference, Dubrovnik, Croatia.
Kurtoglu, T., Swantner, A. and Campbell, M.I. (2010) 'Automating the conceptual design process:"From black box to component selection"', Artificial Intelligence for Engineering Design, Analysis and Manufacturing, vol. 24, no. 01, pp. 49-62.
Nagel, R.L., Vucovich, J.P., Stone, R.B. and McAdams, D.A. (2007) 'Signal Flow Grammar From the Functional Basis', Guidelines for a Decision Support Method Adapted to NPD Processes.
Pahl, G., Beitz, W., Wallace, K. and Council, D. (1984) Engineering design, Springer.
Prabhakar, S. and Goel, A.K. (1998) 'Functional modeling for enabling adaptive design of devices for new environments', Artificial intelligence in Engineering, vol. 12, no. 4, pp. 417-444.
Rihtarsic, J., Zavbi, R. and Duhovnik, J. (2012) 'Application of wirk elements for the synthesis of alternative conceptual solutions', Research in Engineering Design, vol. 23, no. 3, pp. 219-234.
Sarkar, B. (2014) Synthesis of Conceptual Designs, [Online], Available: http://desine.cpdm.iisc.ernet.in:9000 [10 Dec 2014 - Intranet ONLY Access].
Sarkar, B., Chakrabarti, A. and Ananthasuresh, G.K. (2015) 'A New Approach to Conceptual Design Synthesis of Sensors', The Third International Conference on Design Creativity, Bangalore.
Schmitt, G. (1993) 'Case-based design and creativity', Automation in construction, vol. 2, no. 1, pp. 11-19.
Smedley, K. and Cuk, S. (1994) 'Switching flow-graph nonlinear modeling technique', Power Electronics, IEEE Transactions on, vol. 9, no. 4, pp. 405-413.
Srinivasan, V. and Chakrabarti, A. (2009) 'Sapphire--an Approach to Analysis and Synthesis', DS 58-2: Proceedings of ICED 09, the 17th International Conference on Engineering Design, Vol. 2, Design Theory and Research Methodology, Palo Alto, CA, USA, 24.-27.08. 2009, 417-428.
Starling, A.C. and Shea, K. (2005) 'A parallel grammar for simulation-driven mechanical design synthesis', ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 427-436.
Yan, W., Zanni-Merk, C., Cavallucci, D. and Collet, P. (2014) 'An ontology-based approach for using physical effects in inventive design', Engineering Applications of Artificial Intelligence, vol. 32, pp. 21-36.
Zavbi, R. and Duhovnik, J. (2000) 'Conceptual design of technical systems using functions and physical laws', AI EDAM, vol. 14, no. 1, pp. 69-83.