9 DESIGNING NOVEL ARTIFACTS: A NOVEL SYSTEMATIC FRAMEWORK

Srinivasan V* **and Amaresh Chakrabarti†**

**Centre for Product Design and Manufacturing, Indian Institute of Science, Bangalore, India, Tel: +91-(0)80-22933136. E-mail: srinivasan@cpdm.iisc.ernet.in*

†Centre for Product Design and Manufacturing, Indian Institute of Science, Bangalore, India. Tel: +91-(0)80-22932922, Fax: +91-(0)80-23601975. E-mail: ac123@cpdm.iisc.ernet.in

The paper presents a systematic prescriptive framework for designing novel artifacts. The framework integrates activity, outcome and requirement-solution elements and supports conceptual and early embodiment phases. Activity, outcome, requirement and solution represent the human part of problem-solving, properties of artifact at different abstraction levels, characteristics of what the artifact should have at different abstraction levels and means to satisfy requirement(s) at different abstraction levels, respectively. **G**enerate-**E**valuate-**M**odify-**S**elect (**GEMS**), **S**tate change-**A**ction-**P**art-**Ph**enomenon-**I**nput-o**R**gan-**E**ffect (**SAPPhIRE**) and co-evolving requirement and solution are used as activity, outcome and requirement-solution elements. The framework is divided into requirement and solution-synthesis stages. In the requirement-synthesis stage, requirements at different levels including SAPPhIRE are generated, evaluated, modified and selected. In the solution synthesis stage, solutions at different levels of SAPPhIRE are generated, evaluated, modified and selected.

Keywords: Model, Framework, Novel, Activity, Outcome, Requirement, Solution.

1. INTRODUCTION

Designing is a process that spans from the identification of a need to the development of a set of instructions to manufacture at least a solution for the need. Designing is important because it is a means to satisfy human needs. However, designing is complex because of the simultaneous interaction, within and among elements: artifact, tools, people, process, organization and environment.[1]

The aim of design research is to develop knowledge in the form of guidelines, methods and tools to improve the chances of producing a successful product.[1] The aim can be realised by understanding the current situation and based on this understanding, creating pathways for an improved situation.

A model is defined here as a description of how designing is done. A framework is defined here as a prescription of how designing can be done in order to improve some of its characteristics. Thus, developing model(s) and framework(s) enables an understanding of a current situation and creation of an improved situation, respectively.

An earlier work in Ref. 2 developed a model of designing by identifying its elements from literature and validated the model using protocol studies of designing sessions to understand how designing is generally done. The work illustrated in this paper is an attempt at improving how designing is currently done by prescribing a framework for designing. This systematic framework — GEMS of SAPPhIRE as requirement-solution, integrates activity, outcome and requirement-solution based designing elements.

2. LITERATURE SURVEY

This section reports significant findings from literature.

Novelty stands for newness and originality. Creativity is important because new ideas improve quality of products in a competitive market[3] and help increase products' price[4] and hence earn more market

share. For engineering products, novelty is taken as one of the measures of creativity[5] and so it can be argued that if creativity is important then novelty cannot be neglected.

Physical laws and effects (henceforth, referred together as effects) are the principles of nature governing a change.[6] Effects are important in designing because they aid in designing novel artifacts.[7] However, effects were discovered by scientists for explanation of phenomena rather than for designing artifact(s) that embody these phenomena.[7] As a consequence, effects have not been adequately used in designing and this is verified empirically in Ref. 2 using protocol studies of designing sessions.

Conceptual design is a phase of designing that determines principle(s) of solution(s).[8] It places the greatest demands on designer(s), offers maximum scope for striking improvements and the most important decisions are taken during this phase.[9] However, being an early phase of designing, it is associated with open-endedness and thus, needs to be supported.

Activity is defined here as human problem-solving phases in designing. Activities performed on a design problem play a significant role on the success of the end product.[10] Therefore, it becomes important to identify the activities in designing. In Ref. 2, based on a number of activity-based models from literature, generate, evaluate, modify and select are identified as the activities in designing and a **GEMS G**enerate-**E**valuate-**M**odify-**S**elect) model is proposed (Table 1 & Figure 1). The proposed model is validated empirically using protocol studies of designing sessions to check if all the instances in the designing sessions could be represented by the model and vice-versa.

Outcome is defined here as a property of a product at an abstraction level, that is used to specify the product at that abstraction level. The outcomes are used in designing the product and therefore, decide the nature of the end product. The *SAPPhIRE model of causality* (Figure 2) apart from using **E**ffects uses other constructs: **S**tate change, **A**ction, **P**arts, **Ph**enomenon, **I**nput, and o**R**gan (Table 2) and is developed in Ref. 6 to explain the behavior of natural and engineered systems. The constructs of the model constitute the three views of describing an artifact: function, behavior and structure. *Action, state change* and *input* constitute the *function*, *phenomenon* and *effect* constitute the *behavior*, *organ* and*parts* constitute the *structure*. Since function, behavior and structure can be used for both, explaining causality and designing, it can be argued that the SAPPhIRE model should also be able to support designing (Figure 3) apart from explaining causality. This is tested empirically in Ref. 2

Table 1. Definition of activity constructs.[2]

Construct	Definition
Generate	An activity which brings a design outcome into a particular episode
Evaluate	An activity which judges the quality, importance, amount or value of a design outcome in that episode
Modify	An activity which brings about a change in the design outcome in that episode
Select	An activity which chooses the design outcome in that episode

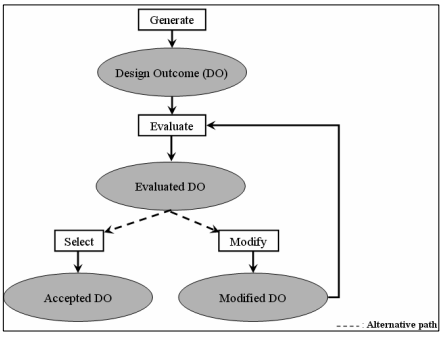

Figure 1. GEMS activity model.[2]

Table 2. Definition of activity constructs.[6]

Construct	Definition
Action	Abstract description or a high-level interpretation of a change of state or changed state or creation of an input
State change	Attributes and the values of attributes that define the properties of a system at a given time during its operation
Input	Energy, material or information requirements that are required for a physical effect to be activated; interpretation of energy/material parameters of a change of state in the context of an organ
Phenomenon	Set of potential changes associated for a given physical effect, organ and input
Effect	Principles of nature governing a change
oRgan	Structural context necessary for a physical effect to be activated
Parts	Set of physical components and interfaces constituting the system and its environment of interaction

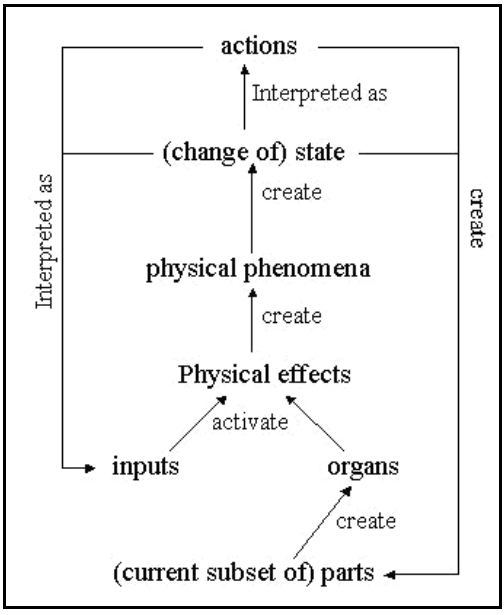

Figure 2. SAPPhIRE model of causality.[6]

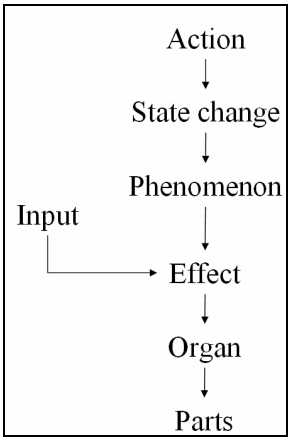

Figure 3. SAPPhIRE model of designing.

using protocol studies of designing sessions by checking if the instances in the designing sessions can be represented using the SAPPhIRE constructs and vice-versa. Less phenomenon-, effect- and organ-level descriptions and higher action- and part-level descriptions are reported. This model of designing supports conceptual and early embodiment phases only.

Requirement is a description of how an artifact should be at a particular abstraction level. Requirements are critical in designing because they initiate a design task and their fulfillment serves as a success criterion.[11] Solution is a means to satisfy requirement(s) and therefore, also critical in designing. A co-evolving model of requirement and solution is proposed in Ref. 2 by observing similar patterns from literature. This is also validated empirically in Ref. 2 using protocol studies of designing sessions and the following patterns are reported: requirement-requirement, requirement-solution, solution-requirement, and solution-solution.

A model of designing is reported in Ref. 2 by integrating the activity (GEMS), outcome (SAPPhIRE) and requirement-solution (coevolving) models and is empirically validated using protocol studies of designing sessions. The model reflects the current way of designing and some key observations are: lower number of solutions and requirements generated, evaluated, modified and selected at phenomenon-, effect and organ-level and higher number of solutions and requirements generated, evaluated, modified and selected at action- and part-level. These observations point the fact that designers are not equally at ease with all the SAPPhIRE constructs, especially phenomena and effects. This could hamper the chances of designing a novel artifact. Based on the above observations, a framework for designing — GEMS of SAPPhIRE as requirement-solution is proposed in Ref. 2 as a support for design for novelty, by prescribing GEMS to be carried out in sufficient detail at all the levels of SAPPhIRE for both requirements and solutions. The novelty part of the framework is contributed by the use of its outcomes. However, the framework has so far only been broadly described and lacks details to support designing.

To establish the relationship between novelty and the outcomes (SAPPhIRE) of solutions, an empirical study is carried out using a different set of protocol studies in Ref. 12 to check if there is a relationship and also to determine the degree of relationship between novelty and the SAPPhIRE constructs. A relation is found and the correlation value decreases with decrease in the abstraction levels, emphasizing the significance of the use of higher abstraction-level constructs especially for novelty.

To summarise, *novelty* is an important aspect to be considered while designing. *Conceptual* design is an early phase and needs more attention. *Activity* (*GEMS*), *outcomes* (*SAPPhIRE*) and *requirement-solution* (coevolving) are important elements for designing and the use of SAPPhIRE constructs may promote novelty. A *framework* for designing novel artifacts that integrates activity, outcome and requirement-solution elements can address the above issues and such a framework needs to be developed because it has not been done before. This paper details the development of such a framework.

3. RESULTS

The framework is divided into two stages: requirement- and solution-synthesis. In the requirement- and solution-synthesis stages, requirements and solutions respectively at different levels of outcome are generated, evaluated, modified and selected.

3.1. Requirement Synthesis Stage

This stage starts by identifying all kinds of requirements including the ones at different levels of SAPPhIRE and culminates with a finalized set of requirements by going through the following sequence of steps:

(a) *Generate* all possible *requirements* for a given design problem to be solved. Classify the requirements into one of the SAPPhIRE constructs and rate the requirements based on their level of importance.
(b) *Evaluate* the *requirement(s)* to check that they do not contradict and are feasible within the scope of the project.
(c) If the requirements do not contradict and are feasible, *select* them.
(d) If the *requirements* contradict or not feasible, *modify* them and repeat step (b).

3.2. Solution synthesis stage

This stage starts by identifying solution(s) at the action-level and culminates with a finalized set of solution(s) at the part-level. The solution synthesis stage is sub-divided into action-, state change-, phenomenon-, effect-, input-, organ- and part-level synthesis stages.

3.2.1. Action-level Solution Synthesis

(a) Based on the selected action-level requirement(s), *generate* an *action*-level *solution*.

(b) *Evaluate* the *action*-level *solution* against the selected action-level requirement(s) to check that it does not violate.
(c) *Select* the *action*-level *solution* if it does not violate.
(d) *Modify* the *action*-level *solution* if it violates and goto (b).
(e) Repeat steps (a)–(d) to get more solution(s) at the action-level.

3.3. State Change-level Solution Synthesis

(a) Based on a selected action-level solution, *generate* a *state change*-level *solution*.
(b) *Evaluate* the *state change*-level *solution* against: action-level solution from which it was generated and selected state change-level requirement(s), to check that it does not violate both.
(c) *Select* the *state change*-level *solution* if it does not violate both.
(d) *Modify* the *state change*-level *solution* if it violates even one of them and goto (b).
(e) Repeat steps (a)–(d) to get more state change-level solution(s) from:
(i) the action-level solution and (ii) all other action-level solution(s).

3.4. Phenomenon-level Solution Synthesis

(a) Based on a selected state change-level solution, *generate* a *phenomenon*-level *solution*.
(b) *Evaluate* the *phenomenon*-level *solution* against: state change-level solution from which it was generated and selected phenomenon-level requirement(s), to check that it does not violate both.
(c) *Select* the *phenomenon*-level *solution* if it does not violate both.
(d) *Modify* the *phenomenon*-level *solution* if it violates even one of them and goto (b).
(e) Repeat steps (a)-(d) to get more phenomenon-level solutions from: (i) the state change-level solution and (ii) all other state change-level solution(s).

3.5. Effect-level Solution Synthesis

(a) Based on a selected phenomenon-level solution, *generate* an *effect*-level *solution*.
(b) *Evaluate* the *effect*-level *solution* against: phenomenon-level solution from which it was generated and selected effect-level requirement(s), to check that it does not violate both.
(c) *Select* the *effect*-level *solution* if it does not violate both.
(d) *Modify* the *effect*-level *solution* if it violates even one of them and goto (b).
(e) Repeat steps (a)-(d) to get more effect-level solutions from: (i) the phenomenon-level solution and (ii) all other phenomenon-level solution(s).

3.6. Input- and Organ-level Solution Synthesis

(a) Based on a selected effect-level solution, *generate* an *input*- and the corresponding *organ*-level*solution*.
(b) *Evaluate* the *input*-/*organ*-level *solution* against: effect-level solution from which they are generated and selected input-/organ-level requirement(s), to check that they do not violate both.
(c) *Select* the *input*-and *organ*-level *solutions* if both of them do not violate their evaluation criteria.
(d) *Modify* the *input*- and *organ*-level *solution* even if one of them violates their evaluation criteria and goto (b).

3.7. Part-level Solution Synthesis

(a) Based on a selected organ-level solution, *generate* a *part*-level *solution*.
(b) *Evaluate* the *part*-level *solution* against: organ-level solution from which it was generated and selected part-level requirement(s), to check that it does not violate both.
(c) Select the part-level solution if it does not violate both.
(d) Modify the part-level solution if it violates even one of them and goto (ii).

(e) Repeat steps (a)–(d) to get more part-level solutions from: (i) the organ-level solution and (ii) all other organ-level solution(s).

3.8. Pictorial Representation

The framework is represented in Figure 4 and the following notations are used:

G: Generate, **E**: Evaluate, **M**: Modify, **S**: Select; **a**: action, **s**: state change, **i**: input, **ph**: phenomenon, **e**: effect, **i**: input, **r**: organ, **p**: part; **re**: requirement, **so**: solution; x[y(z)]: An activity 'x' of requirement or solution 'y' at the outcome level 'z'. For e.g. G[re(a)] means **G**eneration of **re**quirement at the **a**ction-level, S[so(p)] means **S**election of **so**lution at the **p**art-level etc.; E[y(z) (y'(z')]: An evaluation of solution y which is at a level z against requirement or solution y' which is at a level z'. For example, E[so(p) (re(p)] means **E**valuation of **so**lution at the **p**art-level against **re**quirement at the **p**art level, E[so(p) (so(r)] means **E**valuation of **so**lution at the **p**art-level against **so**lution at o**r**gan-level; Y: Yes, N: No

4. DISCUSSION

This section discusses: (a) some rationale behind the steps in the framework, (b) situations that could arise while using the framework and (c) on how these maybe tackled:

The requirement synthesis stage requires that all kinds of requirements be synthesized. But in practice, requirements may also be synthesized during the solution synthesis stage. As long as the requirement(s) synthesized during the solution synthesis stage do not cause any contradictions with requirements synthesized earlier, or are at an abstraction level that is lower than the level of solutions synthesized until then, these can be included in the list of requirements without any violation of earlier requirements and solutions. However, if new requirements that violate either of these two are generated, then the already selected requirements and solutions must be evaluated against the new requirements to check and ensure that they comply with these. This will involve additional iterations, may lead to modifications of already selected requirements and solutions, which amount to an inefficient usage of design time. By following the above approach there is a potential advantage in synthesizing requirements before solutions.

In the requirement synthesis stage it might not be possible to classify each requirement precisely using a SAPPhIRE construct since the SAPPhIRE constructs at the part level are not currently worked out in detail to cater for requirements at the later design phases. However, requirement synthesis being the first stage in designing demands that all kinds of requirements including those that pertain to the later stages of designing be synthesized. However, only those requirements that can be classified using the SAPPhIRE constructs are currently considered later in the framework during solution synthesis. It is not necessary that each SAPPhIRE construct should have a requirement. It may also be possible that there are no requirements at some levels of SAPPhIRE.

Level of importance is a variable that determines the relative importance of a requirement among a set of requirements. The following are some factors based on which the level of importance can be assigned:

(a) Abstraction level: A requirement has a higher level of importance if it is at a higher abstraction level compared to that of other requirements. This is because constructs at a higher abstraction level create more impact than those at a lower level.
(b) Source: A requirement can come from the problem statement or from designer(s). Requirements from problem statement may be assigned a higher level of importance. This is because these requirements may have to be compulsorily addressed.
(c) Designer's discretion: The level of importance can be assigned by the designer based on his/her experience.
(d) Hybrid: The level of importance can be assigned after taking the above three factors into consideration. For instance, level of importance can be a measure of a product of the level of importance of the level of abstraction, source and designer's discretion.

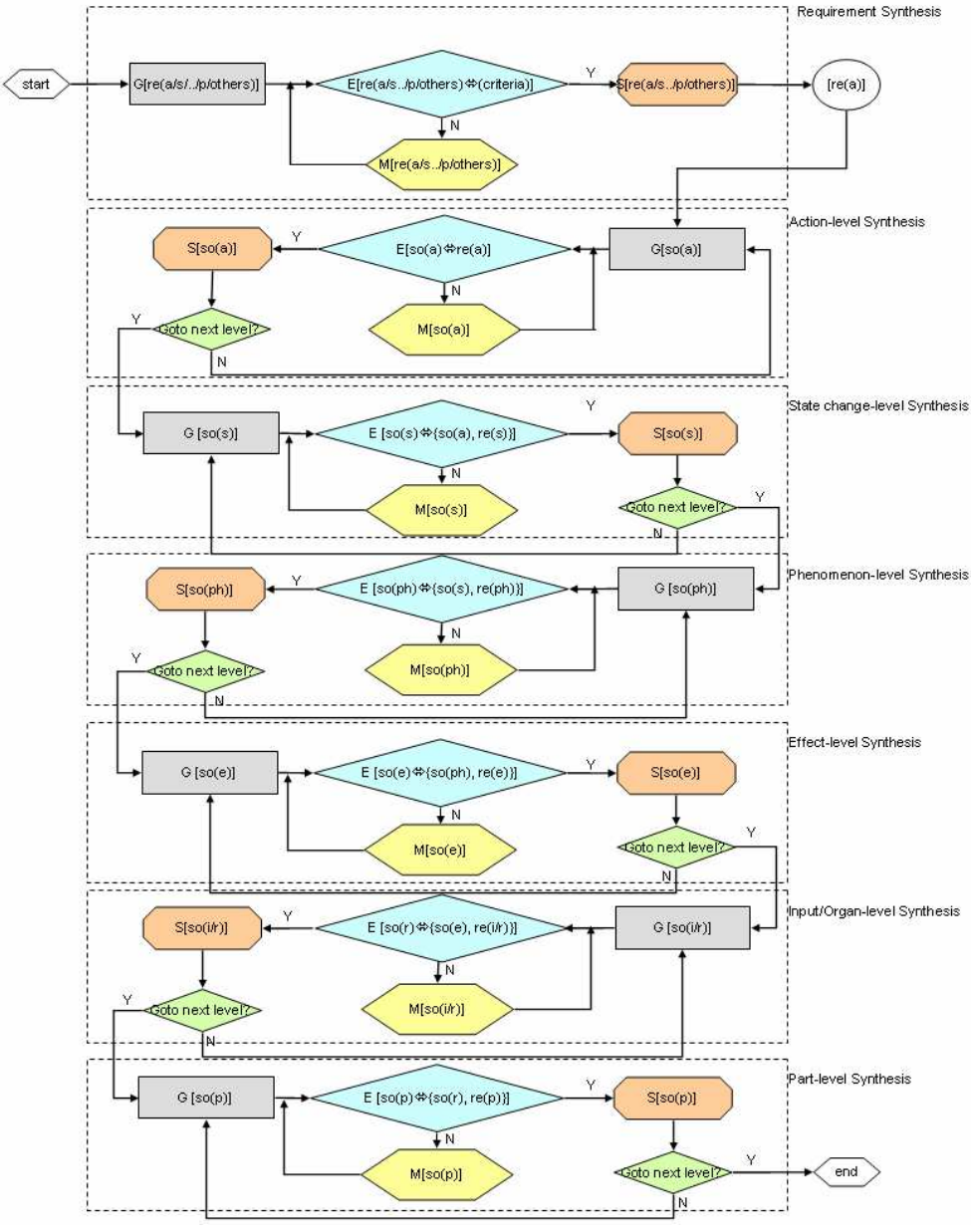

Figure 4. Pictorial representation of the framework.

In requirement synthesis guidelines for modifying the requirements are proposed as:

(a) Case 1: If there is a contradiction between two requirements, the one with a lower level of importance maybe modified first, and only if the contradiction still exists then the one with a higher level of importance may be modified. The level of importance can change when a requirement gets modified.

(b) Case 2: If there is a contradiction between more than two requirements, the requirement that is involved in most number of contradictions may be modified first. The requirements have to be arranged in descending order of number of contradictions, first requirement in the order is modified and the number of contradictions has to be checked if it has reduced. If the number of contradictions has not reduced, then the second requirement has to be modified (without modifying the first) in the order and checked again. The above steps have to be repeated until no contradictions exist. If more than a requirement has the same number of contradictions then the requirement with the least level of importance among them maybe modified first.
(c) Case 3: The design problem is solved with requirement contradictions (similar to TRIZ approach), so that solutions that satisfy such requirements have a greater chance for novelty.

The framework has two kinds of requirements: (1) Solution-neutral requirements: These are synthesized during the requirement synthesis stage, and (2) Solution-specific requirements: These are synthesized during the solution-synthesis stage and correspond to selected solutions at an abstraction level which are used for generating solutions at the immediately next lower abstraction level. A natural coevolution is observed between solution-specific requirements and solutions during the solution synthesis stage but one cannot rule out the coevolution between solution-neutral requirements and solutions.

In the solution synthesis stage, if there is no requirement at a particular abstraction level, then it becomes difficult to evaluate solutions at that level. However, a designer's domain experience with related products helps overcome related obstacles.

In the evaluation of a solution designers may face difficulty while evaluating a solution simultaneously against multiple requirements. An approach would be to start by arranging the requirements in the descending order of level of importance and start by evaluating against the first requirement and if there is no violation, proceed by evaluating the solution against the next requirement and so forth. However, if a solution does not match a requirement, it has to be modified and the evaluation has to start from the first requirement.

In the solution synthesis stage, input-level and organ-level solutions are synthesized together. This is because an effect requires an input and set of organs for its activation. For a given effect satisfying a phenomenon, there is only a unique combination of input and organs. Note that the number of input-level solutions equals the number of effect-level solutions.

During the solution synthesis stage it is possible that a solution cannot be modified or the modified solution does not conform to requirements even after finite iterations of modification and evaluation. An approach to tackle this would be to reject this solution and generate another solution. The key point to be considered is that there has to be at least one solution selected at any level so that the solution(s) at next level can be developed. However, if not even a single solution can be selected due to non-conformance with the requirement(s), then the requirement(s) that the solution does not conform to can be modified and the earlier procedures in the framework repeated. In cases where not even a single solution can be generated at a level by a human, external assistance (experienced designer, expert system, computer-support) maybe sought.

At each level of solution synthesis it is difficult to define the (maximum) number of iterations (modification and evaluation) for a solution before it gets selected or rejected. It is also difficult to define the minimum/maximum number of requirements and solutions to be synthesized at each stage of requirement and solution synthesis. A requirement at action-level and a solution at each level are the basic necessities.

In both requirement- and solution-synthesis stages, no explicit criteria for evaluation are mentioned because it restricts the framework to a particular domain. But we believe that the framework can support designing of any generic novel artifacts. A few general criteria like feasibility, ergonomics, manufacturability, (higher) efficiency/effectiveness, (low) cost etc maybe also be considered and these criteria would get reflected only when the design is detailed into a tangible product.

Overall, the framework so far primarily only details about 'what' is to be done during designing. However, more details pertaining to 'how' it could be done would provide a more comprehensive support.

5. SUMMARY AND FUTURE WORK

This section summarizes the contents of the paper and gives directions for future work:

(a) A prescriptive framework for designing — GEMS of SAPPhIRE as requirement-solution that integrates activity, outcomes and requirement-solution has been developed for designing novel artifacts.
(b) The framework currently supports only conceptual and early embodiment phases and needs to be extended to support the later phases of designing.
(c) The framework is intended as a support for designing novel products but needs a comprehensive evaluation to check if it can support novelty.

REFERENCES

[1] Blessing, L., Chakrabarti, A. and Wallace, K. (1995). A Design Research Methodology, *Intl Conf on Engg Design (ICED95)*.
[2] Srinivasan, V. and Chakrabarti, A. (2008). Design for Novelty – A Framework?, *Intl Design Conf (DESIGN08)*.
[3] Molina, A., Al-Ashaab, A. H., Ellis, T. I. A., Young, R. I. M., and Bell, R. (1995). A Review of Computer Aided Simultaneous Engineering Systems, *Res Eng Des*, 7:38–63.
[4] Ottosson, S. (1996). Boosting Creativity in Technical Development, WDK 24: Engineering Design and Creativity, *Proc of Workshop EDC*, W.E. Eder (ed.), Pilsen, pp. 35–39.
[5] Shah, J. J., Smith, S. M. and Vargas-Hernandez, N. (2003). Metrics for Measuring Ideation Effectiveness, *Design Studies*, 24: 111–134.
[6] Chakrabarti, A., Sarkar, P., Leelavathamma, B. and Nataraju, B. S. (2005). A Functional Representation for Aiding Biomimetic and Artificial Inspiration of New Ideas, *AIEDAM*, 19: 113–132.
[7] Murakoshi, S. and Taura, T. (1998). etc. Research on the Systematization of Natural Laws for Design Support, *IFIP Workshop on Knowledge Intensive CAD*, pp. 141–160.
[8] Pahl, G. and Beitz, W. (1996). *Engineering Design: A Systematic Approach, Springer*.
[9] French, M. (1999). *Conceptual Design for Engineers*, Third Edition, Springer, London.
[10] Cooper, R. G. (1990). New Products: What Distinguishes The Winners?, *Research Technology Management*, pp. 27–31.
[11] Nidamarthi, S. (1999). Understanding and Supporting Requirement Satisfaction in the Design Process, PhD Thesis, Gonville and Caius College, Cambridge, UK.
[12] Srinivasan, V. and Chakrabarti, A. (2008). Novelty-SAPPhIRE Relationship: An Empirical Study, *Intl Conf on Trends in Product Lifecycle Modelling, Simulation and Synthesis*.