ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (642-649), Month: April - June 2015, Available at: www.researchpublish.com

Optimised Solutions for Issues in Design of a Search Program

[1]Roshni R, [2]Mahesh.K Kaluti

[1]PG Student, Dept. of CSE, AIET,
[2]Asst. Professor, Dept. of CSE, AIET

Abstract: **Task structure for issues in design of search programs is done by analyzing a general class of methods; the task structure is constructed by identifying a range of methods for each task. For each method, the knowledge needed and the subtasks that it sets up are identified. Most of the subtasks are not really specific to design as such. The analysis shows that there is no one ideal method for design and good design problem solving is a result of recursively selecting methods based on a number of criteria, including knowledge availability There is no crystal clear criteria for issues in search of a design process, Analysis of problems gives us enormous solutions and inclearity, selecting best is the high time problem faced by designers. We can reach the perfection of selecting solution for design of search programs By Appling practicality, reliability, desirability, scalability to the thought process while implementing programs to the desired task.**

Keywords: **Taskstructure, Reliability, Scablity, Desirability.**

1. INTRODUCTION

Search programs are used to find required information, and results should be displayed, solving problems regarding design of a search program is a daunting process, it is integrated with task, each task contains many subtasks which have many methods.

A research study has popped out a survey result stating that structure of task has to be given maximum priority for solving issues The designed task structure contains optimised subtask ,among many subtask ,the method is integrated in the structure should be applayble, the knowledge required for above process should be available within the task structure, designing a system involves a systematic step to achieve an particular goal that is system design ,during design there are lot of steps like requirements, specification, deadlines, verification validation and modification these process Has to be carried out in an efficient way to get the optimised results of design process of a search programs, issues arises at every method so ,the problems are noted at every at phase and solutions are given to them so that at final stage the design of search programs are optimised and the system should happen to give maximum results, it even involves removing bugs at every stage, the models are designed considering all the facts of designing process and solving the issues in the design process, the model has a systematic step which gives solution more quickly than expected.

These models are enacted with efficient search programs with on issues providing results at stretch.

Factors Needed To Be Considered For The Issues In The Design Of A Search Programs:

Design process is a synthetic task since it involves searching in the group of many optimised solutions, It is a search problem which has multiple constraints, only some task satisfies all the constraints, designers usually stick on to the optimal solution by considering hypothesis artistic creations and scientific theories, The optimised solution can be modified to get adopted for the execution of different task.

2. THE DESIGNED SUBTASK PROVIDE SAFETY

It should provide solutions and perform the task, which is to get the results, we can consider the example of Google search engine which provides the desired result, and the designed search engine programs there by provide reliability, security and the desired results.

We have to pinup functions and constraints. Where function states about the purpose of a design of a search programs, constraints are the restrictions put on to the design of a search programs, Issues in design of a search program should emphasis on functions, constraints relations and components.

A function is a block of code that has a name and has a property, which is reusable and can be executed from different points. The design of a search programs should know the main goal behind the design.

Components are nothing but an building block of a system or is an identified part of a larger programmer or construction, While design of a task we should connect with the sub task that is between the components constraints is nothing but limitations or restriction.

The design should contain the above constraint, Design the such program so that it stays committed to the specified constraints throughout the design process, Relations are nothing but a process in which two thing get connected, the relation should be efficiently mentioned between task, sub task and components for designing efficient programs.

Criterias to Be Considered While Solving Issues in Design of a Search Program:

Design task need to consider all the task, sub task, methods should be combined to give an optimised result. We can give the solutions to the design process by giving a set of components and the relationship among them, comparing a group of components and relationship amongst them will give a solution.

Solutions should match all the criteria before. We may need to refashion the issues of search programs, During design stage, only few functions and constraints might be accusable ,the required solution should be added sequentially.

A bit of design process, needs to be refashioned, where some functions and constraints need to be redefined to achieve the task. We can even elaborate over design process for search programs to desired task.

For every single task we should have some procedure for accomplishing it. These operators have to be oriented in a crystal clear way to accomplish over intention. If we have to multiply two numbers by using the method of recursion, the method of recursion where it can be used as a subtask.

We can go for algorithms self defined step to set of operations for design programs with no issues. We have to find out a resolution for the accomplishment of target. Else, we have to modify the goal. Rectify the constraints, subtask, methods associated with it.

Disintegrating The Design To Solve The Problem Associated:

The larger problem for design should be broken into smaller subproblems, the optimised disintegration method should be chosen by using available knowledge. In search process it should be conducted during design process, after that disintegrate the problem into sub problems and integrate that as final report, if we follow the above ideology the design problem will be lessened.

3. DESIGN TACTICS FOR DESIGN

Design skills ,creativity and design process all are combined to achieve an optimised design for search programs, while design process we should set some goal at base level later we can add cup or expand our set design goals at the mid of the design phase called as refinement phase, it leads to provide the optimised design. We have set an objective for each and every sub problem; we should focus towards design for solutions issues in design of a search program. a case study is an research method involving in-depth and detailed examination of a subject study and its contextual conditions. We have to look into case studies of the already designed system which are nearer to goal, after choosing the particular design we need to do some refinements so that it satisfies the given goal, it in turn saves time, money and gives optimised results.

Constraint Satisfaction:

Constraint satisfaction is the process of finding a solution to set constraints that impose conditions and variable must satisfy condition. It will be put forward for any design process; the design process should be flexible enough to get adapted to the constraints.

4. VERIFICATION AND VALIDATION

Verification is done at the starting of the development process; it includes receiving's, walkthroughs to evaluate the plan code. It should be done properly at the development phase to get the best results.

Verification and Validation example is also given just below to this table.

Verification	**Validation**
1. Verification is a static practice of verifying documents, design, code and program.	1. Validation is a dynamic mechanism of validating and testing the actual product.
2. It does not involve executing the code.	2. It always involves executing the code.
3. It is human based checking of documents and files.	3. It is computer based execution of program.
4. Verification uses methods like inspections, reviews, walkthroughs, and Desk-checking etc.	4. Validation uses methods like black box (functional) testing, gray box testing, and white box (structural) testing etc.
5. **Verification** is to check whether the software conforms to specifications.	5. **Validation** is to check whether software meets the customer expectations and requirements.
6. It can catch errors that validation cannot catch. It is low level exercise.	6. It can catch errors that verification cannot catch. It is High Level Exercise.
7. Target is requirements specification, application and software architecture, high level, complete design, and database design etc.	7. Target is actual product-a unit, a module, a bent of integrated modules, and effective final product.
8. Verification is done by QA team to ensure that the software is as per the specifications in the SRS document.	8. Validation is carried out with the involvement of testing team.
9. It generally comes first-done before validation.	9. It generally follows after **verification**.

Suppose we have the specifications related to the project than by checking that specifications without executing to see whether the specifications are up to the mark or not is what we have done in verification. Similarly Validation of the software is done to make sure that the software always meets the requirements of the customer by executing the specifications of the design process, Considering example, if you are developing a computer performing tic tac toe game then users are the end users they are the people who has to give the requirements we should verify the requirements and break the task or components into smaller components and in each phase we have to verify everything to get the result.

Techniques or Methods of Verification and Validation:

(A) Methods of Verification

1. Walkthrough

2. Inspection

3. Review

(B) Methods of Validation

1. Testing

2. End Users

Summary:

1) Verification and Validation both are necessary and complementary.

2) Both of them provide its own sets of Error Filters.

3) Each of them has its own way of detect out the errors left in the software.

Lots of people use verification and validation interchangeably but both have different meanings Verification process describes whether the outputs are according to inputs or not, and Validation process describes whether the program designed is accepted by the system or not.

5. REQUIREMENTS AND SPECIFICATION

Requirements is about mentioning the customer's needs of a particular system ,suppose if we want to design an system we need certain factors as in purpose, time limit, inputs ,outputs and most importantly meeting the deadlines. It should be clarified and documented to generate the corresponding specifications. While it is a common tendency for designers to be anxious about starting the design and implementation, discussing requirements with the customer is vital in the construction of safety-critical systems in artificial intelligence. For activities in this first stage has significant impact on the downstream results in the system life cycle. For example, errors developed during the requirements and specifications stage may lead to errors in the design stage. When this error is discovered, the engineers must revisit the Send specifications to fix the problem. This leads not only to more time wasted but also the possibility of other requirements and specifications errors. Many accidents are traced to requirements flaws, incomplete implementation of specifications, or wrong assumptions about the requirements. While these problems may be acceptable in non-safety-critical systems, systems cannot tolerate errors due to requirements and specifications. Therefore, it is necessary that the requirements are specified correctly to generate clear and accurate specifications.

There is a distinct difference between requirements and specifications. A requirement is a condition needed by a user to solve a problem or achieve an objective. A specification is a document that specifies, in a complete, precise, verifiable manner, the requirements, design, behaviour, or other characteristics of a system, and often, the procedures for determining whether these provisions have been satisfied. For example, a requirement tic tac toe game is two users should be present and a board should be there to do the moments,. The specification for this requirement would include technical information about specific design aspects. Another term that is commonly seen in books and papers is requirements specification which is a document that specifies the requirements for a system or component. It includes functional requirements, performance requirements, interface requirements, design requirements, and development standards, specification of tic-tac toe game employed in a system is the way in which the game plated and an utility Value which gives the end result . So the requirements specification is simply the requirements written down on paper, TESE requirements and specification should be mentioned properly to solve the issues in the design process.

6. SIMULATION

Simulation is the process of copying real word process ,we have to create an dummy model and check for the design model, we have to proceed after we get the needs of the users ,design verification and validation process involves checking whether our design has meet the specification and meets its intended purpose, if our design process fails analysis involves to wait for something in order to discover the meaning of essential features, we have to break down components into essential features, if a part of the design process fails to give the required output ,we should have the idea about the area to go ahead and check out the cause for the failure of the system, we should check its precondition and post condition for the design process

7. MODIFICATION

The changes that adopts to limits ,qualifies or reconstructs something to a new end ,we have to apply an backtracking method to get an result at faster rate ,sometimes design may fail due to unavoidable reasons ,in that case we may need to change the functional description of subpart to achieve the task. Design Technology is about generating ideas to solve problems that results in a product:

Programs needed for the system may need periodic up gradation of the parts connected to it to achieve the optimisation ,Skills are required for up gradation we should know the updated versions of the programs, we should about meeting the deadline performance keeping cost criteria in mind.

- Skills of investigation
- Skills of designing solutions to problems Skills of adopting programs
- Skills in evaluating

The skills needed by designers mainly are Technical Skills includes knowledge about the issues in design of a search programs, and all possible techniques required to accomplish the task. And the associated processes or procedures. These skills relate to the Making stage. This could be called the applied learning stage. Design Skills relating to Investigating, Designing and Evaluating is called thinking stage to accomplish the task.

Designing search programs solutions for the issues is purposeful, systematic, and creative, with many possible solutions. While it is recognised that designer prior learning has equipped them with an ability to identify, manage, and resolve problems, a four part designing model: investigating, designing, producing and evaluating, is provided as a model that can be used.

Design and Technology involves working within Design Briefs or requirements to satisfy human needs and wants for a client or customer with an efficient search programs with minor errors. It is an interactive decision-making process involving thinking, investigating, creating, and devising with continuous evaluation and, often, modification as a result of this evaluation.

Investigating can include activities such as:

- specifying plans and purposes from a need analysis, and identifying tasks;
- establishing outcomes;
- researching, analysing, and communicating information;
- searching for solutions;
- Critical analysis of programs characteristics.

Designing:

Designing can include activities such as,

- Devising flexible, imaginative, innovative and enterprising outcomes.
- Preparing sketches, concept drawings, and working drawings.
- Testing, modifying, and validating ideas.
- Selecting appropriate solutions.

Producing:

Producing can include activities such as:

- Making articles to chosen standards and specifications for programs.
- developing skills and applying them to new situations;
- Controlling quality, televising and using

Procedures, processes, or sequences;

- working alone or collaboratively;
- Using resources, equipment, or materials.

Evaluating:

Evaluating can include activities such as:

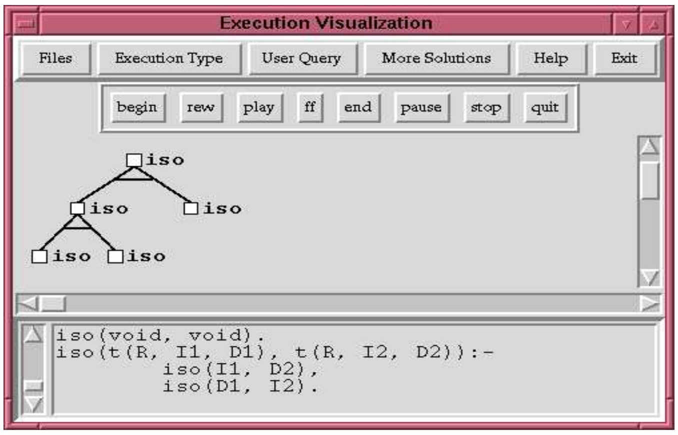
- evaluating how well the requirements of the design brief have been met;
- redeveloping or improving plans, ideas, or procedures;
- preparing reports on outcomes;
- considering personal achievement and learning against criteria;
- involving other people in the evaluation process;
- communicating with people about outcomes and expectations;
- critiquing outcomes of prototypes against similar retail products;

Taking action based on what have been learned ability, safety, and cost

Modification is needed after verification and validation, since many issues will be solved at this stage but modification is all about upgradability of the programs after issues being solved we need to modify certain parts of design so that the issues should not arrive again.

8. CASE STUDTY OF DIFFERENT OPTIMISED DESIGN MODELS

Model 1: A Small Execution Tree as shown by apt



```
iso(void, void).
iso(t(R, I1, D1), t(R, I2, D2)):-
        iso(I1, D2),
        iso(D1, I2).
```

Figure 1: A small execution tree, as shown by APT

The programmed search part of CLP execution is similar in many ways to that of LP execution. The visualization of this part of (C) LP program execution traditionally takes the form of a direct representation of the search tree, whose nodes represent calls, successes, redoes and failures| i.e., the events

Which take place during execution Classical LP visualization tools, of which the Transparent Prolong Machine (TPM [EB88]) is paradigmatic, are based on this representation In particular, the TPM uses an augmented AND-OR tree (AORTA), in which AND OR branches are somewhat compressed and take up less vertical space, but the information conveyed by it is basically the same as in a normal AND-OR tree.

It is true that in many CLP programs (and especially in those using Finite Domains) the control part has less importance than in LP, since the bulk of the time in such programs is spent in equation solving, but the programmers designed the model by implementing the search programs by looking into all issues, as shown in Figure 1the design model will have user query and more solutions if the CLP node execution fails we can take help of the expertise and et an solution, user query will keep track of all query by looking at this interface the expertise will give best and more solutions.

Model 2 for A search with branching and depth both equal to the tree:

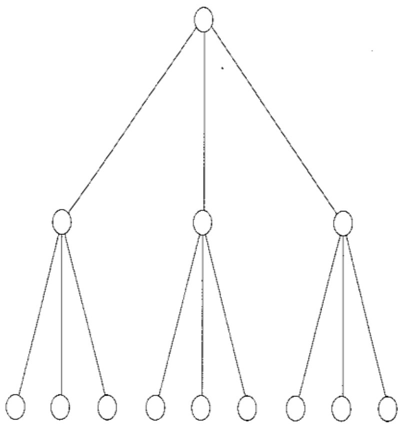


Figure 2: A search with branching and depth both equal to the tree

A search space is typically represented by a tree, the nodes in the tree are the search states and are connected by operators, a brute force tree searches the tree without having the knowledge. there bare two technology namely depth first and breadth first search ,in case of BFS all the nodes are at one level of a tree, before proceeding onto the next level, it examines the tree node by node, requires exponential storage, in case of depth first search it attempts a single line of processing from he root to the way to find the solution, selects one child root then goes for another child of the root till the solution is found, the main advantage of DFS is storage is linear, if we compare it to the traditional method that is brute force then BFS and DFS top lines it.

Model 3: Applications 2 Sussman 894

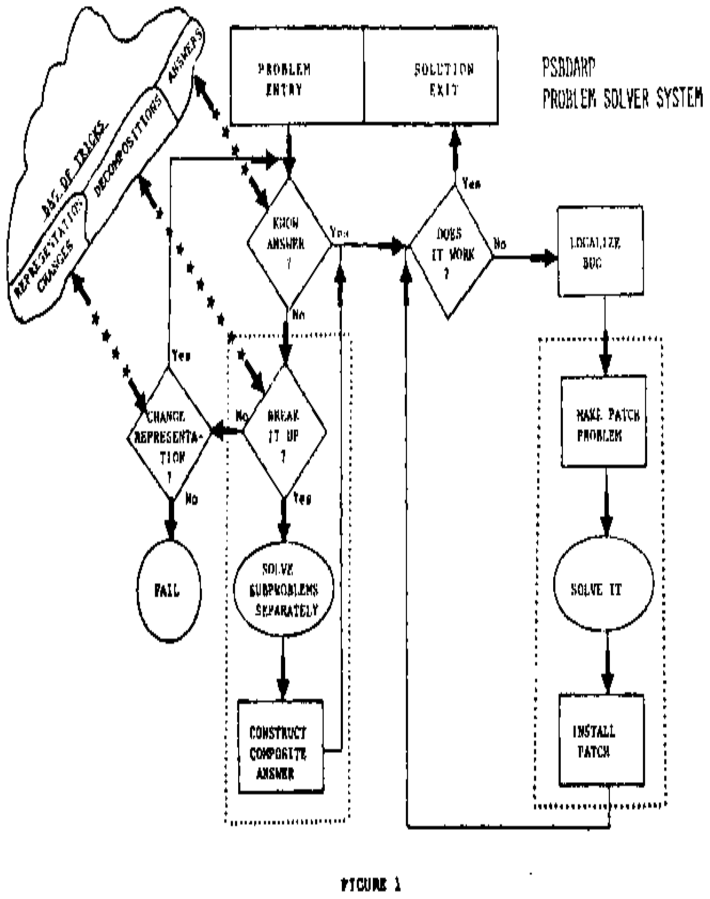


Figure 3: Applications-2 : Sussman 841

Figure 3 displays the structure of a PSBDARP problem solver. When the problem solver is given a problem it first checks its Answer Library to determine if there is an answer available whose pattern of applicability matches the problem statement. If so, the proposed answer is tested make sure that it really works and if it passes the test it is returned as the answer

Of the problem posed. But suppose the answer is not immediately available. The problem solver next examines a set of problem decompositions to see if any are appropriate for breaking the problem into more manageable chunks. If so, the problem solver remembers the decomposition rule chosen and recursively calls itself to solve each sub problem separately. If This is possible the solutions returned are combined according to the decomposition rule used to break the problem up and the resulting proposal is sent off to be tested. If there are no decomposition rules available which match the problem statement, the problem solver next checks to see if there are any changes, of representation which can be applied to the problem statement to put it into a form more amenable to solution If so, the problem is considered in terms of the alternate representation.

If no representation changes are appropriate, the problem solver has failed on this problem and reports its failure. A failure may cause backtracking and search. In PSBDARP there is a specific phase of the solution process where debugging knowledge is applied. This knowledge is relatively domain independent and is concerned with notions of causality, technology and simultaneous constraints. The debugging phase is far more concerned with the structure of the plan produced by the decomposition phase than it is with the goal that evoked the plan. For example, localization of a bug in an electrical circuit or in a computer program may involve such strategies as "tracing" — examination of the conditions at various module boundaries to determine how the expected conditions compare with those that actually occur in the process. The Figure 3 clearly shows the optimised form of an search program where the solutions for problems are calculated decomposing components, then sub problem separately each phase, we can trace up the errors by solving it by using track and backtrack method.

9. CONCLUSION

There are lot of issues regarding the design of a search process, the system performance depends on the requirements specification and factors needed to be considered for the issues in the design of a search programs, starting from design facts, design consideration, modification, verification and validation, we have to divide each steps into sub steps,the problems should be solved separately in each phase to get the solution.

Saving justifications for the intermediate results of computation has other merits. It is very difficult to debug programs containing large amounts of knowledge. The complexity of the interactions between the "chunks" of knowledge makes it difficult to ascertain what is to blame when a bug manifests itself. A program which can explain the reasons for its beliefs is more convincing when right, and it is easier to debug when wrong, further research is required to gather the information about the solutions for the issues in design process.

REFERENCES

[1] Balzer, R. 1981. Transformation Implementation:An Example. IEEE Transactions on Software Engineering SE-7: 3–14.

[2] Brown, D. C., and Chandrasekaran, B. 1989. DesignProblem Solving: Knowledge Structures and ControlStrategies. San Mateo, Calif.: Morgan Kaufmann.

[3] Bylander, T. C. 1988. A Critique of Qualitative Simulationfrom a Consolidation Point of View. IEEESystems, Man, and Cybernetics 18(2): 252–268.Chandrasekaran, B. 1989. Task Structures, KnowledgeAcquisition, and Learning. Machine Learning 4:339–345.

[4] Chandrasekaran, B. 1986. Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocksfor Expert System Design. IEEE Expert 1(3): 23–30.

[5] Chandrasekaran B., and Narayanan, N. H. 1990.Integrating Imagery and Visual Representations. InProceedings of the Twelfth Annual Conference of theCognitive Science Society, 670–677.

[6] Hillsdale, N.J.:Lawrence Erlbaum.Clancey, W. J. 1985. Heuristic Classification. ArtificialIntelligence 27(3): 289–350.

[7] Dixon, J. R.; Simmons, M. K.; and Cohen, P. R.1984. An Architecture for Application of ArtificialIntelligence to Design. In Proceedings of the Twenty-First Design AutomationConference, 634–640.Washington, D.C.: IEEE Computer Society.

[8] Forbus, K. D. 1988. Qualitative Physics: Past, Present,and Future. In Exploring Artificial Intelligence,eds. H. E. Shrobe and the American Association for Artificial Intelligence, 239–296. San Mateo, Calif.

[9] Morgan Kaufmann.Friedland, P. 1979. Knowledge-Based ExperimentalDesign in Molecular Genetics. In Proceedings ofthe Sixth International Joint Conference on Artificial Intelligence, 285–287. Menlo Park, Calif.: InternationalJoint Conferences on Artificial Intelligence.

[10] Goel, A. 1989. Integration of Case-Based Reasoningand Model-Based Reasoning for Adaptive DesignProblem Solving, Ph.D. diss., Dept. of Computerand Information Science, The Ohio State Univ.

[11] Goel, A., and Chandrasekaran, B. 1989a. FunctionalRepresentation of Designs and Redesign ProblemSolving. In Proceedings of the Eleventh InternationalJoint Conference on Artificial Intelligence,1388–1394.

[12] Menlo Park, Calif.: International JointConferences on Artificial Intelligence. Goel, A., and Chandrasekaran, B. 1989b. Use ofDevice Models in Adaptation of Design Cases. InProceedings of the DARPA Workshop on Case-Based Reasoning ed. K. Hammond, 100–109. San Mateo,Calif.: Morgan Kaufmann. Goel, V., and Pirolli, P. 1989.

[13] Notion of Generic Design within Information-ProcessingTheory: The Design Problem Space. AI Magazine10(1): 18–38.Hammond, K. 1989. Case-Based Planning:ViewingPlanning as a Memory Task. Boston: Academic.Hayes-Roth, B. 1985. A Blackboard Architecture.