International Journal of Database Theory and Application
Vol.7, No.3 (2014), pp.49-62
http://dx.doi.org/10.14257/ijdta.2014.7.3.06

Architecture Synthesis Process for Scheduler Concept Based On Design Algebra with Object Oriented Design Methods

P. Rajarajeswari[1], Dr. A. Ramamohanreddy[2] and Dr. D. vasumathi[3]

[1]*Researchscholar in CSE, JNTUH, Assistant professor in Madanapalle Institute of Technology and Science*
[2]*Professor in Department of Computer science and Engineering, SV University, Tirupathi*
[3]*Professor in Department of Computer science and Engineering, Jntuh, Hyderabad*
[1]*perepicse@gmail.com,* [2]*ramamohansvu@yahoo.com,* [3]*vasukumara_ devara@yahoo*

Abstract

Synthesis analysis is the effective process for solving problems in Software engineering. We provide description for scheduler concept by using synthesis analysis. We presented implementation of architecture design alternatives for scheduler concept based on design algebra concept. During the process of alternative space analysis different alternative solutions are searched and also evaluating the different design solutions based on quality criteria or constraints. This is more effective processes for designing the different systems based on quality criteria and constraints. In this paper balancing design alternatives for this concept with object oriented design methods.

Keywords: *Software engineering, Synthesis analysis, design algebra, object oriented design methods.*

1. Introduction

Synthesis analysis is required for a broad perspective. Software solutions are used for solving Technical problems in software engineering To explicitly reason about the concepts of problem solving, a model for problem solving that may be used for analyzing various problem-solving activities is presented. Synthesis model is used for analyzing problem solving in software. Synthesis analysis is used for solving software architecture design problems and also improves the quality of software systems.

Our previous studies have shown that that the state-of-the-art architecture design approaches are not aligned with an explicit synthesis process. Usually in software architecture design processes during the problem analysis, solution domain analysis and alternative space analysis is either implicit or not well-defined. In the problem solving process, First of all identify technical problems, and those problems are divided into sub problems. Solve each sub problem individually that are to be combined for the complete solution of the problem. In this process analysis is performed on technical problems, solution domain and alternative space. Extract the relevant Solution abstractions from the given problem domains.

This paper can be organized as follows. We presented a model for synthesis for in Section 2. In Section 3 we provide synthesis process for scheduler concept. We implemented different architecture design alternatives based on design algebra concept and also balancing design space alternatives in Section 4. We concluded in Section 5.

2. A Model for Synthesis

ISSN: 2005-4270 IJDTA

Synthesis technique is used for the implementation of problem-solving process. The following fig presents a software architecture design which is based on synthesis analysis. Conceptual model consists of two parts: one of the parts is Solution Definition and second part is Solution Control. Concepts and functions are used in each part of the design. Solution abstractions are identified and also defined by Solution Definition phase. Quantitative method, measurement, optimization techniques/Heuristic rules and improvement of the selected solution abstractions are given in the Control phase for Solution.

Solution Definition

Client requirements are to be represented in the concept of Specification of requirements. Technical Problem concept refers to the given problem which is to be solved. Define the process by searching the problem solutions with the help of functions that can be used for architecture development.

Problem can be decomposed into number of sub problems.

Identify the relevant sub problem from the main problem that is to be represented by the Select function. The concept relevant information for solution represents the knowledge which provides solution of the sub-problem.

Examine function represents the relevant information by using searching process based on a given problem.

Solution Abstraction concept gives the retrieved solution from the relevant information. The function retrieve represents the process for extracting the solution abstractions from the relevant information. To provide extracted solution specifications by using Specification of Solution Structure concept .The function Specify represents the process for specifying the solution abstraction.

The concept Architecture Description represents the design details of Architecture description so far.The function form represents the refinement of the overall-design description with the concept.

Solution Structure Specification.

The function Discover represents the process of discovering new sub-problems when new solution abstractions are retrieved from the relevant information.

The function effect represents the process of refining the requirement specification from the results of the architecture design specifications for architecture description.

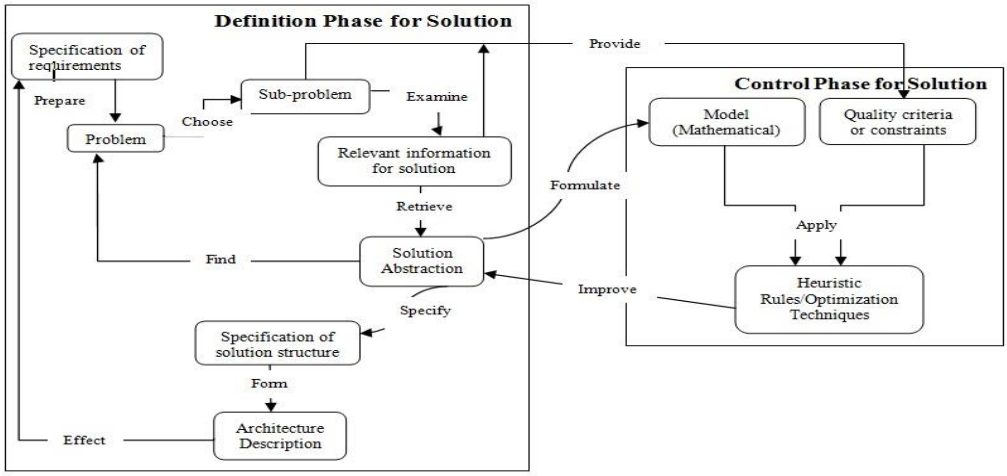

Figure 1. Problem Solving Process for Software Architecture Design based on Synthesis Analysis

3. Synthesis Process for Scheduler Concept

It consists of three processes:

Analysis on Technical Problem: It defines the problem definition and divides the problem into number of sub-problems which are to be solved. In this analysis problem can be defined by taking the some of the parameters such as quality criteria and the constraints are imposed on the problem formulation. Mathematical symbols, formulas and equations are used for expressing quality criteria and constrains,

Well-defined problem statement is defined by the quantification of the quality criteria and the constraints. The objectives are ordered from higher level to lower-level. From the given specification, the engineers can easily applied different alternatives to calculate the feasibility of the end-product.

Analysis on Solution Domain: It examines the relevant information from the solution domain and its modeling in Order to solve the problems. Some of the handbooks are referred to get relevant information for the given problem.

Analysis on Alternative Space: It provides the alternative space generation and their alternatives.

Alternatives are retrieved from the relevant literature survey. Evaluation of alternatives can be performed with the help of mathematical modeling. Alternatives are evaluated by using mathematical modeling. Artifact can be described in terms of mathematical expressions. Different mathematical models may be used to provide various aspects of the similar design alternatives.

3.1. Case study: Example

In the following we provide description part for the concept of the Scheduler system architecture:

We define the problem structural diagram for the concept of Scheduler and also added the sub-problems P1.1 Concurrency control P1.3Deadlock management. The conceptual architecture of Scheduler consists of three sub-concepts: concurrency mechanisms, concurrency strategies, Deadlock management.

The concept defines the Concurrency approach by receiving, delaying or aborting the incoming operations. It addresses the problem Concurrency mechanisms may be basically through locking, time stamping and serializibility. The concept Concurrency Strategy also addresses the problem P1. There is a difference is in between pessimistic and optimistic schedulers. An optimistic scheduler is focused on delay operations, whereas pessimistic scheduler avoids these aborts the operation sooner delays. The concept deadlock management addresses is concerned by the problem P.2 *i.e.*, deadlock management which is focused on the detection of performance failures such as deadlocks.

Alternative Design Space Analysis: The alternative space analysis is used to define the set of possible different design solutions which can be derived based on the conceptual software architecture. The Alternative Design Space Analysis aims to depict this space and consists of identifying the design alternatives for each concept and provides description for the design alternatives based on the constraints. The following table gives the different alternatives for each concern of the concept.

Example: Let us consider the alternatives for the concept of scheduler architecture. We define the space by providing the table with column header. Sub concepts are represented by the column headers. An instance of the sub-concept is represented by the table entry.

Table 1. Alternatives of the Sub Concepts of Scheduler

S.No.	A. Concurrency mechanisms	B. Concurrency strategies	C. Deadlock management
1	Locking	pessimistic	Deadlock Detector
2	Time stamping	optimistic	Deadlock avoidance
3	Serializibility		Deadlock prevention

4. Implementation of Architecture Design Alternatives Based On Design Algebra Concepts Outcomes

Design algebra technique is used for providing the formal structure of a software system. Sets and operations are used in design algebra. Manipulation of relations can also be possible by using Relational algebra. Design Algebra can be considered as one of the application of relational algebra that can be used for modeling design spaces and also balancing design alternatives. Design Algebra introduces operations on design spaces.

Concept of Design Space

Design Space:

The concept of design space is used to represent the different alternatives for implementing a conceptual model of a software system.

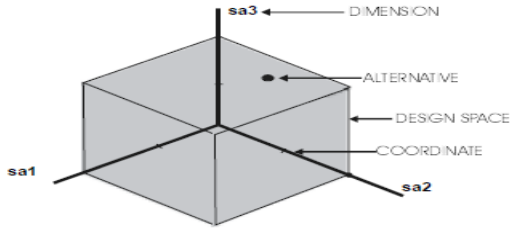

Figure 2. Design Space

Design space: Set of design alternative solutions is to be found for a given problem that may be represented in a multi-dimensional design space.

Dimensions: Solution abstractions are to be represented as dimensions in an orthogonal direction of Design space .Alternative set of elements can be considered as dimensions.

Coordinates: An instance of alternatives is representing by an element of a dimension.

We define a design space as a multi-dimensional space from which the set of alternatives for a given design problem can be derived. We may define a design space for every concept in a solution domain. The design space is spanned by an independent set of dimensions. The dimensions are represented by the sub concepts of the concept in the solution domain.

Consider, for example, the concept Scheduler that can formally be described as follows.

$$\mathbf{M_{Scheduler} = (Cm, Cstr, Dm)}$$

Here, Cm, CStr and Dm represent the sub-concepts Concurrency mechanisms, Concurrency Strategy and Deadlock management, respectively. A design space for Scheduler consists of three dimensions, which are represented in the above table by these sub-concepts. Every dimension has a set of coordinates that are elements of a coordinate set. In design algebra, the coordinate set represents a property set that represent various properties that may be assigned to the sub-concepts. An example of a property set may be the concepts of the object-oriented model. In the object-oriented model, a concept can be represented either as a class, an operation or an attribute. Therefore we may define the property set Object as follows:

$$\mathbf{P_{Object} = (CL, OP, AT)}$$

The degree of a dimension represents the total numbers of the properties. A design alternative represents a point in the design space.Design Algebra defines the design space as function spaces that map concepts to properties. This is described as follows:

$$\mathbf{S_{ObjectSchedule}r::M_{Scheduler} \rightarrow P_{Object}}$$

The following figure shows the graphical representation of an example of a design space that utilizes the sub-concepts of Scheduler as dimensions and the coordinates of these dimensions are the properties of the set $\mathbf{P_{Object}}$. The design space is named $\mathbf{S_{ObjectScheduleraccomplishment.}}$

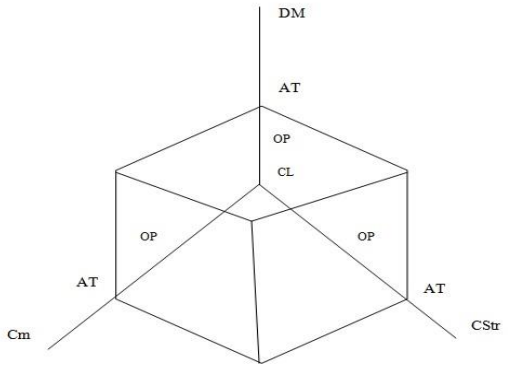

Figure 3. Design Space of Scheduler with Object as Property Set

This function describes all theoretically possible alternatives in mapping the concepts to properties from the property set Object. Figure 7 shows a graphical representation of this design space function.

The number of design alternatives it introduces can be derived as follows:

Number Of Alternatives (Sobjectschduler) =size (Pobject) size (Mscheduler) $=3^3=27$

Since this design space has 27 points this implies that there are in total 27 theoretically possible design alternatives within this space.

M domain consisting of sub concepts such as c1,c2,……..,cn and p1,p2……pm are the properties in property set Pproperty.The design space Sproperty Domain can be defined in formal way given below.

Sproperty Domain ::Mdomain→Pproperty

Calculate number of alternatives by using the given formula.

Number of alternatives $(S_{\text{property Domain}})=\text{size}\ (P_{\text{property}})^{\text{size}(M_{\text{Domain}})}=m^n$

For example (*((Cm,CL) (CStr,CL) (Dm,CL))* which gives the selection of the sub-concepts of *Scheduler* as classes in an alternative design space.

Design space: Set of design alternative solutions is to be found for a given problem that may be represented in a multi-dimensional design space.

Dimensions: Solution abstractions are to be represented as dimensions in an orthogonal direction of Design space .Alternative set of elements can be considered as dimensions.

Coordinates: An instance of alternatives is representing by an element of a dimension.

We define a design space as a multi-dimensional space from which the set of alternatives for a given design problem can be derived. We may define a design space for every concept in a solution domain. The design space is spanned by an independent set of dimensions. The dimensions are represented by the sub concepts of the concept in the solution domain.

Consider, for example, the concept Scheduler that can formally be described as follows.

$$\mathbf{M_{Scheduler} = (Cm, Cstr, Dm)}$$

Here, Cm ,CStr and Dm represent the sub-concepts Concurrency mechanisms, Concurrency Strategy and Deadlock management, respectively. A design space for Scheduler consists of three dimensions, which are represented in the above table by these sub-concepts.

Every dimension has a set of coordinates that are elements of a coordinate set. In design algebra, the coordinate set represents a property set that represent various properties that may be assigned to the sub-concepts. An example of a property set may be the concepts of the object-oriented model. In the object-oriented model, a concept can be represented either as a class, an operation or an attribute. Therefore we may define the property set Object as follows:

$$\mathbf{P_{Object} = (CL, OP, AT)}$$

The degree of a dimension represents the total numbers of the properties. A design alternative represents a point in the design space.

Design Algebra defines the design space as function spaces that map concepts to properties. This is described as follows:

$$\mathbf{S_{ObjectSchedule}r::M_{Scheduler}\rightarrow P_{Object}}$$

The following figure shows the graphical representation of an example of a design space that utilizes the sub-concepts of Scheduler as dimensions and the coordinates of these dimensions are the properties of the set $\mathbf{P_{Object}}$. The design space is named $\mathbf{S_{ObjectScheduleraccomplishment.}}$

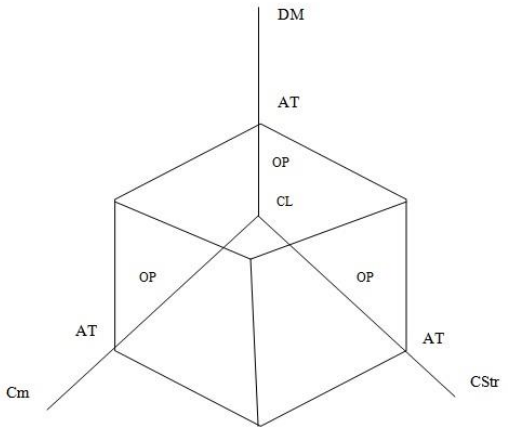

Figure 3. Design Space of Scheduler with Object as Property Set

This function describes all theoretically possible alternatives in mapping the concepts to properties from the property set Object. Figure 7 shows a graphical representation of this design space function.

The number of design alternatives it introduces can be derived as follows:

Number Of Alternatives (Sobjectschduler) =size (Pobject) size (Mscheduler) $=3^3=27$

Since this design space has 27 points this implies that there are in total 27 theoretically possible design alternatives within this space.

M domain consisting of sub concepts such as c1,c2,……..,cn and p1,p2……pm are the properties in property set Pproperty.

The design space Sproperty Domain can be defined in formal way given below.

Sproperty Domain ::Mdomain→Pproperty

3

Calculate number of alternatives by using the given formula.

Number of alternatives ($S_{property\ Domain}$)=size ($P_{property}$)$^{size(M_{Domain})}$=m^n

For example (*((Cm,CL) (CStr,CL) (Dm,CL))* which gives the selection of the sub-concepts of *Scheduler* as classes in an alternative design space.

$S_{ObjectScheduler}$ =

{((Cm,CL) (Cstr,CL) (Dm,CL)), ((Cm,CL) (Cstr,CL) (Dm,OP)), ((Cm,CL) (Cstr,CL)(Dm,AT)),

((Cm,CL) (Cstr,OP) (Dm,CL)), ((Cm,CL) (Cstr,OP) (Dm,OP)), ((Cm,CL) (Cstr,OP) (Dm,AT)),

((Cm,CL) (Cstr,AT) (Dm,CL)), ((Cm,CL) (Cstr,AT) (Dm,OP)), ((Cm,CL) (Cstr,AT)(Dm,AT)),

((Cm,OP) (Cstr,CL) (Dm,CL)), ((Cm,OP) (Cstr,CL) (Dm,OP)), ((Cm,OP) (Cstr,CL) (Dm,AT)),

((Cm,OP) (Cstr,OP) (Dm,CL)), ((Cm,OP) (Cstr,OP) (Dm,OP)), ((Cm,OP) (Cstr,OP) (Dm,AT)),

((Cm,OP) (Cstr,AT) (Dm,CL)), ((Cm,OP) (Cstr,AT) (Dm,OP)), ((Cm,OP) (Cstr,AT)(Dm,AT)),

(Cm(,AT) (Cstr,AT) (Dm,CL)), ((Cm,AT) (Cstr,AT) (Dm,OP)), ((Cm,AT) (Cstr,AT)(Dm,AT)),

((Cm,AT) (Cstr,OP) (Dm,CL)), ((Cm,AT) (Cstr,OP) (Dm,OP)), ((Cm,AT) (Cstr,AT) (Dm,AT))

((Cm,AT) (Cstr,AT) (Dm,CL)), ((Cstr,AT) (Cstr,AT) (Dm,OP)), ((Cm,AT) (Cstr,AT)(Dm,AT))

}

4.1. Balancing Design Space Alternatives

Description of the design alternatives for a design concept can be considered as one of the way that is to be introduced in the Design space
To reduce the design space by using the following techniques

1. Selection criteria or manual selection is used for Selection of a sub-space.

2. Non-feasible alternatives based on constraints are not considering for the Elimination of a sub-space.

3. Applying the first or second technique through heuristic rules or methods for Heuristics-based selection and/or elimination.

Selection of a sub-space

Selection of a sub-space from a design space is distinguished in three selection techniques.
1. Direct selection by Software engineer

2. Selection criteria is based on conditional specifications

3. Matrix-based selection for choosing design alternatives

Software engineer directly selects alternatives from the Direct selection which is a simple technique in the design space.

One may use selection criteria based on conditional specifications for larger design spaces. It can be described by using the general form which is given below.

{((Cm,CL) (Cstr,CL) (Dm,CL)), ((Cm,CL) (Cstr,CL) (Dm,OP)), ((Cm,CL) (Cstr,CL)(Dm,AT)),

((Cm,CL) (Cstr,OP) (Dm,CL)), ((Cm,CL) (Cstr,OP) (Dm,OP)), ((Cm,CL) (Cstr,OP) (Dm,AT)),

((Cm,CL) (Cstr,AT) (Dm,CL)), ((Cm,CL) (Cstr,AT) (Dm,OP)), ((Cm,CL) (Cstr,AT)(Dm,AT)),

((Cm,OP) (Cstr,CL) (Dm,CL)), ((Cm,OP) (Cstr,CL) (Dm,OP)), ((Cm,OP) (Cstr,CL) (Dm,AT)),

((Cm,OP) (Cstr,OP) (Dm,CL)), ((Cm,OP) (Cstr,OP) (Dm,OP)), ((Cm,OP) (Cstr,OP) (Dm,AT)),

((Cm,OP) (Cstr,AT) (Dm,CL)), ((Cm,OP) (Cstr,AT) (Dm,OP)), ((Cm,OP) (Cstr,AT)(Dm,AT)),

(Cm(,AT) (Cstr,AT) (Dm,CL)), ((Cm,AT) (Cstr,AT) (Dm,OP)), ((Cm,AT) (Cstr,AT)(Dm,AT)),

((Cm,AT) (Cstr,OP) (Dm,CL)), ((Cm,AT) (Cstr,OP) (Dm,OP)), ((Cm,AT) (Cstr,AT) (Dm,AT))

((Cm,AT) (Cstr,AT) (Dm,CL)), ((Cstr,AT) (Cstr,AT) (Dm,OP)), ((Cm,AT) (Cstr,AT)(Dm,AT)) }

Selection criteria based on conditional specifications:

To provide the list of all alternatives and process of Scanning of all these alternatives requires more time. To overcome this problem to give a condition that specifies alternatives which are chosen by Software Engineer.
Design Space ::{ concept→property Set |(condition)}

Concept elements are mapped to the elements of Property Set based on the logical expressions constraints.

sub concept →propeŧty , where sub concept ε concept and property→ property Set.

- condition1∧condition2 , that evaluates to true if both condition1 and condition2 evaluate to true
- condition1∨condition 2 , that evaluates to true if either condition1 or condition2 or both of them evaluate to true.
- ¬condition, that is true if condition is false; it is false if condition is true

Example:

Define a reduced sub-space of S RObjectScheduler by using the following formula. This formula can be used for the selection of alternatives from the design space in which the concurrency mechanisms is a class or concurrency mechanisms is a operation.

$$S\ RObjectScheduler ::\{ M_{Scheduler} \rightarrow P_{Object} \mid (Cm \rightarrow CL) \vee (Cm \rightarrow OP) \}$$

Normally 18 alternatives are possible:

{ ((Cm,CL) (Cstr,CL) (Dm,CL)), ((Cm,CL) (Cstr,CL) (Dm,OP)), ((Cm,CL) (Cstr,CL) (Dm,AT)),

((Cm,CL) (Cstr,OP) (Dm,CL)), ((Cm,CL) (Cstr,OP) (Dm,OP)), ((Cm,CL) (Cstr,OP) (Dm,AT)),

((Cm,CL) (Cstr,AT) (Dm,CL)), ((Cm,CL) (Cstr,AT) (Dm,OP)), ((Cm,CL) (Cstr,AT) (Dm,AT)),

((Cm,OP) (Cstr,CL) (Dm,CL)), ((Cm,OP) (Cstr,CL) (Dm,OP)), ((Cm,OP) (Cstr,CL) (Dm,AT)),

((Cm,OP) (Cstr,OP) (Dm,CL)),((Cm,OP) (Cstr,OP) (Dm,OP)), ((Cm,OP) (Cstr,OP)(Dm,AT)),

((Cm,OP) (Cstr,AT) (Dm,CL)), ((Cm,OP) (Cstr,AT) (Dm,OP)), ((Cm,OP) (Cstr,AT) (Dm,AT)) }

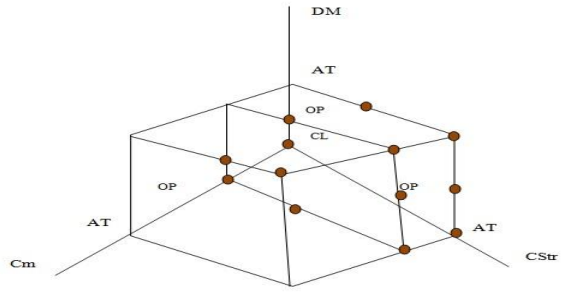

Figure 4. Alternatives of Object Scheduler

This sub-space selection can be visualized.

Reduction of Design space Sobjectscheduler

SObjectScheduler ::{ MScheduler → PObject | ((Cm → CL)∨ (Cm →OP))∧ (Dm→CL) ∨(Dm→ OP))}

The following design alternatives possible.

{((Cm,CL) (Cstr,CL) (Dm,CL)), ((Cm,CL) (Cstr,CL) (Dm,OP)), ((Cm,CL) (Cstr,OP)(Dm,CL)), ((Cm,CL) (Cstr,OP) (Dm,OP)),)) ((Cm,CL) (Cstr,AT) (Dm,CL)), ((Cm,CL) (Cstr,AT)(Dm,OP)), ((Cm,OP) (Cstr,CL) (Dm,CL)), ((Cm,OP) (Cstr,CL) (Dm,OP)), ((Cm,OP) (Cstr,OP)(Dm,CL)), ((Cm,OP) (Cstr,OP) (Dm,OP)), ((Cm,OP) (Cstr,AT) (Dm,CL)), ((Cm,OP) (Cstr,AT) (Dm,OP)), }

The number of alternatives may be computed in advance so that in case this number is too large,the designspace may be furher reduced by providing more selection criteria.

Assume that software engineer is further interested in selecting only the alternatives in which concurrency mechanism is represented as either a class or an operation.For this the following expression needs to be specified.

. SObjectScheduler ::{ MScheduler → PObject |((Cm → CL)∨ (Cm →OP))∧ (Cstr →CL)V(Cstr→ OP))}

- Note that this now results in the following 12 alternatives:
- { ((Cm,CL) (Cstr,CL)(Dm,CL)), ((Cm,CL) (Cstr,CL) (Dm,OP)), ((Cm,CL) (Cstr,CL) (Dm,AT)),
- ((Cm,CL) (Cstr,OP) (Dm,CL)), ((Cm,CL) (Cstr,OP) (Dm,OP)), ((Cm,CL) (Cstr,OP) (Dm,AT)),

- ((Cm,OP) (Cstr,CL) (Dm,CL)), ((Cm,OP) (Cstr,CL) (Dm,OP)), ((Cm,OP) (Cstr,CL) (Dm,AT)),
- ((Cm,OP) (Cstr,OP) (Dm,CL)), ((Cm,OP) (Cstr,OP) (Dm,OP)), ((Cm,OP) (Cstr,OP) (Dm,AT)) }
- We can graphically represent this reduced design space as shown by shaded area in the given diagram.

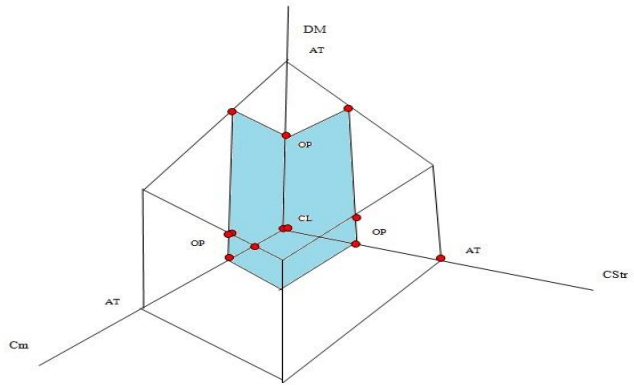

Figure 5. Representing Reduced Design Space

Another technique is matrix-based selection can be used for design alternatives. This technique lists the elements of the design space alternatives in a matrix as opposed to listing each design space alternative. This list is a Cartesian product of the concept and the property set:

Cobjectscheduler=Mscheduler x Pobject

$$\begin{bmatrix} (Cm,CL) & (Cstr,OP) & (Dm,AT) \\ (Cm,CL) & (Cstr,OP) & (Dm,AT) \\ (Cm,Cl) & (Cstr,OP) & (Dm,AT) \end{bmatrix}$$

=(Cm, Cstr , Dm)X(CL,OP,AT)

Alternatives can be composed based on the selection of a tuple from each column. Selection criteria can be done using logical connectives. Generation of design space alternatives can be generated through Combination of all valid row combinations.

Elimination of a sub-space:

Consider for example condition-based selection, , in which one selects alternatives using a selection Expression. By negating the selection expression, it becomes an elimination expression, for

$$S_{Rscheduler}::\{Mscheduler \rightarrow \quad Pobject \mid \neg ((Cstr \rightarrow AT) \wedge (Dm \rightarrow AT))\}$$

This expression eliminates the alternatives in which the Cstr or Dm maps to an Attribute. The logical condition expression for elimination can of course be combined with selection conditions.

Heuristics-based selection and/or elimination

The reduction of a design space can be supported by the use of heuristics. What heuristics can be used depends on the problem domain. For the property set $P_{Object,}$ introduction of heuristic rules can be performed with the help of object-oriented analysis and design methods, that is described below.

For example. An elimination technique or selection can be applied based on heuristic rules. This can be described as follows:

IF <condition> THEN <consequent>

A few examples describing the use of heuristic rules (taken from [Rumbaugh *et al.*, 1991]) for selection and elimination can be found below:

IF an entity is relevant

THEN select the entity as a class (CL)

IF an entity is a transient event

THEN eliminate it as an operation (OP)

A heuristics-based selection on relevancy of scheduler could be expressed as follows:

$$S\ RObjectscheduler ::\{Mscheduler \rightarrow Pobject \mid relevant(scheduler) \Rightarrow (scheduler \vdash> CL)\}$$

5. Conclusion

- Design space concept is used for different implementations for software system.
- It allows for the software engineer to make the properties of design concepts explicit where needed – for instance the object-oriented language constructs to use.
- The design space reduction operations are introduced in case of Design Algebra.
- The software engineer can able to select the alternative from a manageable number of design alternatives.
- Automated reduction of design alternatives is possible with this method.

References

[1] Agre, G. P. The concept of problem, Educational Studies, vol. 13, pp. 121-142, 1982.
[2] Alexander, C. Notes on the Synthesis of Form, Harvard University Press, Cambridge, MA, 1964.
[3] Archer, L.B., Systematic Method for Designers, The Design Council, London, also reprinted in [Cross 84], 1965.
[4] Braha, D., & Maimon, O. The Design Process: Properties, Paradigms, and Structure. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 27, No. 2, March 1997
[5] Brown, D.C., & Chandrasekaran B. Design Problem Solving. London: Pitman, 1989.
[6] Budgen, D. Software design, Addison-Wesley, 1994
[7] Chomsky, N., On certain formal properties of grammars, Information and Control 2,2(1959), 137-167, 1959.
[8] Coyne, R.D., Rosenman, M.A., Radford, A.D., Balachandran, M., & Gero, J.S. Knowledge-based design systems, Addison-Wesley, 1990.
[9] Cross, N. Developments in Design Methodology, Wiley & Sons, 1984.
[10] Dijkstra, E. W., "Structured Programming," Software Engineering Techniques, Buxton, J. N., and Randell, B., eds. Brussels, Belgium, NATO Science Committee, 1969.
[11] Hunt, E., Problem Solving, in: Thinking and Problem Solving., ed. R.J. Sternberg, pp. 215-232, Academic Press, 1994.
[12] Maher, M.L. Synthesis and evaluation of preliminary designs, in: Artificial Intelligence in Design, J.S. Gero, Ed. New York: Springer-Verlag, 1989.
[13] Arend, E. van der. Design of an Architecture for a Quality Management Push Framework. MSc thesis, Dept. of Computer Science, University of Twente, 1999.
[14] Barghouti, N.S., & Kaiser, G.E. Concurrency Control in Advanced Database Applications, ACM Computing Surveys, Vol. 23, No. 3, September, 1991.
[15] Bernstein, A., & Goodman, N. Concurrency Control in Distributed Database Systems, ACM Transactions on Database Systems, 8(4): 484-502, 1983.

Authors

P. Rajarajeswari received the B. Tech. from S.V University, Sri Venkateswara University, Tirupathi, Andhra Andhra Pradesh.M. Tech degree in Computer Science from JNT University, Hyderabad and Pursuing Ph.D in Computer Science and Engineering, JNT University Hyderabad Andhra Pradesh, dia. Presently She is working as Assistant Professor of Computer Science and Engineering in Ins Madanapalle Institute of Technology and Science, Madanapalle, Chitoor(Dt), Andrapradesh. Her rhas 8 years of and Teaching experiences. She is a life member of IAENGand IACSIT. Her research interests include Software Architecture, Software Engineering and Data Mining. She has 8 international publications and 8 International conference Publications at International and 6 National level workshops.

Dr. A. Rama Mohan Reddy received the B. Tech. from JNT University, Hyderabad in 1986, M. Tech degree in Computer Science from National Institute of Technology in 2000 Warangal and Ph. D in Computer Science and Engineering in 2008 from Sri Venkateswara University, Tirupathi, Andhra Pradesh, India. He worked as Assistant Professor, Associate Professor and Presently working as Professor of Computer Science and Engineering, Sri Venkateswara University College of Engineering. He has 28 years of Industry and Teaching experience. Currently guiding twelve Ph. D scholars. He is life member of ISTE and IE. His research interests include Software Architecture, Software Engineering and Data Mining. He has 15 international publications and 15 international conference Publications at International and National level.

Dr. D.Vasumathi received the B. Tech. from JNT University, Hyderabad, M. Tech degree in Computer Science and Engineering from JNT University, Hyderabad and Ph. D in Computer Science and Engineering JNT University Hyderabad Andhra Pradesh, India. She worked as Assistant Professor, Presently working as Professor in JNT University, Hyderabad. She has 12 years of Teaching experience. Currently guiding 8 Ph. D scholars. She is life member of ISTE and IE. Her research interests Data Mining, Imageprocessing, Webtechnologies He has 10 international publications and 15 international conference Publications at International and National level.

