Chapter - 9

Conclusions and Future Work

9.1 Overview

This chapter reviews the main contributions of the research study as well as some future lines of investigation which have emerged along the research work. The chapter is organized in the following manner: Section 9.2 summarizes the contributions of the research. In section 9.3, usefulness of the approach developed in the doctoral research work (Thesis) is mentioned. Section 9.4 provides the implementation scheme of the developed unified systems approach. Finally, in section 9.5 future scope of the work is mentioned.

9.2 Contributions of the Thesis

This thesis intends to contribute in several aspects to the field of Component based software engineering. The present work provides a unified methodological framework comprises of graph theoretic systems approach, decision techniques and concurrent engineering principles to assist software development and research industry for complete system design and optimum selection. The following are the significant contributions of the research work:

Systems approach related contributions

- A methodological framework is developed using system methodology and graph theoretic approach to model, analyze and design component based software system. The framework helps in representing CBSS structural information, including its sub-systems, their sub-sub-systems (up to component level) and their interconnections. The methodology consists of the CBSS system structure digraph, the CBSS system permanent matrix, and the CBSS system permanent function. The CBSS digraph is the mathematical representation of the structural characteristics and their interdependence, useful for visual modeling and analysis. The CBSS system permanent matrix converts digraph into another mathematical form. This matrix representation is a powerful tool for storage and retrieval of sub-systems in the computer database and also for computer processing. The CBSS system permanent function is a mathematical model characterizing the

structure of the CBSS product and it also helps one to determine the CBSS system index.

- The permanent function of the CBSS system architecture at a particular level of hierarchy represents all possible combination of its sub-systems. The terms of permanent function not only represent different sub-sets of CBSS system architecture but are also capable of generating large number of alternative design solutions. At each level strategic decisions can be taken for the selection or rejection of components, strategies, procedures and designs. The system's structural characteristic level (i.e. permanent index) of the complete system is computed by calculating the system's structural characteristic level of each sub-system at the lower level and substituting them as diagonal elements of the system permanent matrix at the higher level. The developed systems approach is a very a powerful tool as it is an integrated systems approach. All the sub-systems up to the component level are modeled and evaluated to be used as inputs for diagonal elements at the next higher level and so on. The approach can be used to optimize the design and the development parameters.

- Developed structural coefficients of similarity and dissimilarity and identification sets are useful models to select optimum set of sub-systems up to component level to finally achieve high quality CBSS.

- The developed systems approach is further extended for *usability* modeling and analysis of a software component. Software component specific *usability* characteristic along with sub-characteristics, associated attributes, measures and interactive complexity have been identified. The systems approach comprising of digraph and matrix approach is developed to analyze concurrently *usability* characteristic of a software component based on attributed factors which leads to improvement of the component *usability* both at the designing and development of component. Concept of hypothetical best (*usability*) index and hypothetical worst (*usability*) index is also developed by which improvement on component design, development and selection can be achieved.

- The systems approach is further utilized for developing the *maintainability* index of a software component based on attributed factors for *maintainability*

characteristic. To achieve this, sub-characteristics and associated attributes and their interactions subject to *maintainability* of a component is considered. A *maintainability* digraph in order to analyze the *maintainability* of a component by considering all levels of interactions (inter-intra) is developed. For detailed analysis *maintainability* digraph is transformed to permanent matrix (mathematical form) which retains all information of component's *maintainability*. A unique *maintainability* expression is derived from permanent function which is developed from the permanent matrix. This expression yields component's *maintainability* index. The concept of hypothetical best maintainability index; hypothetical worst *maintainability* index; and component's relative index from hypothetical best *maintainability* index and hypothetical worst *maintainability* index is also developed. Based on this, decision related to selection, optimization, evaluation and ranking of software components, systems designs etc., can be taken as per *maintainability* point of view.

- The developed systems approach is further extended to analyze different failure modes and effects of CBSS which leads to improvement of the CBSS *reliability* at the design stage. The failure modes and effects digraph is constructed to analyze the failure modes and effects of CBSS. The developed permanent function is a useful tool for minimizing the failure modes and effects and it also leads to the characterization, comparison and evaluation of the CBSS as per failure modes and effect point of view. The numerical value of the permanent function is the CBSS failure modes and effects index. This index is a measure of the consequence of the failure modes and effects.

- The developed systems approach is also utilized to compute *reliability* index of *CBSS* based on heterogeneous architecture styles when reliabilities of contributing elements are known. This will help decision makers to identify software component or design which is less or more reliable.

- A systematic methodology based on the developed systems approach is developed for the quality characteristics (sometime called as "*X-bilities*") evaluation of a software component. Quality digraph is created to evaluate the quality of a component by considering relative importance among characteristics. Component

quality is represented as a single expression using permanent matrix (one-to-one mapping to digraph) and permanent function. Quality index is also developed that can be used for the selection and the ranking of candidates.

Indices related contributions

Several indices are developed under this research work:

- *Systems Structure Index* (I_{SS})*:* capable of comparing alternative designs on the basis of number of terms present in the system permanent expression. Using the concept alternative designs can be indentified for similarity and dissimilarity.

- *Usability index* (I_u)*:* The *usability* index is a quantitative measure of the *component*. As *usability* expression, *VPF – u*, considers structural and interactive complexity of sub-characteristics and associated attributes it can be used to generate the measure. Based on the I_u the selection and evaluation of the component can be carried out as per *usability* point of view. To evaluate *VPF – u*, numerical values of U_i (*usability* sub-characteristics) and a_{ij} (interaction between i^{th} and j^{th} sub-characteristics) are required. It is to be noted that using I_u, one can carry out the comparison of two or more than two alternative components available in the market as per *usability* point of view. The designers and developers of the component will also get to know which factor has to be improved so as to increase the overall *usability* of a component.

- *Hypothetical best usability index* (I_{bu})*:* To get hypothetical best *usability* index the level of satisfaction for *usability* sub-characteristics and/or associated attributes has to be set to 5 (maximum level). The resultant value after performing permanent computation is the hypothetical best *usability* index of a component.

- *Hypothetical worst usability index* (I_{wu})*:* To get hypothetical worst *usability* index the level of satisfaction for *usability* sub-characteristics and/or associated attributes has to be set to 1 (minimum level). The resultant value after performing permanent computation is the hypothetical worst *usability* index of a component.

- *Maintainability index* (I_m)*:* The developed *maintainability* index is a quantitative measure of the *maintainability* of a *component*. This index of component needs to take into account concurrently the value of *maintainability* sub-characteristics for the component and their interactions (among sub-characteristics). As *maintainability* expression, *VPF – m*, considers interactive complexity of sub-characteristics and associated attributes in a single *maintainability* expression it can be used to generate the measure. Moreover, all terms of *VPF – m* are positive. Thus, increased values of *VPF – m* terms (entities values) will increase the overall value. Based on the I_m the selection and evaluation of the component can be carried out as per *maintainability* point of view. Assigning qualitative or quantitative values to attributes and interactions, evaluation of index can be accomplished.

- *Hypothetical best maintainability index* (I_{bm})*:* To get hypothetical best *maintainability* index the level of satisfaction for *maintainability* sub-characteristics and/or associated attributes has to be set to 5 (maximum level). The resultant value after performing permanent computation is the hypothetical best *maintainability* index of a component.

- *Hypothetical worst maintainability index* (I_{wm})*:* To get hypothetical worst *maintainability* index the level of satisfaction for *maintainability* sub-characteristics and/or associated attributes has to be set to 1 (minimum level). The resultant value after performing permanent computation is the hypothetical worst *maintainability* index of a component.

- *Relative maintainability index with hypothetical best maintainability index* (I_{mrb})*:* It is defined as the ratio of *maintainability index of a component with hypothetical best maintainability index of a component.* It represents maintainability value of the component as "**%"** of the ideal best value of the index.

- *Relative maintainability index with hypothetical worst maintainability index* (I_{mrw})*:* It is defined as the ratio of *maintainability index of a component with hypothetical worst maintainability index of a component.* It represents

maintainability value of the component as "**%**" of the ideal worst value of the index.

- *Failure index* (I_{cfmea})*:* The concurrent failure mode and effect index (I_{cfmea}) also known as failure index is a quantitative measure of the CBSS. This means it indicates the extent of the consequence in the event of the possible failure mode on component and/or system. As concurrent failure modes and effects expression, *VPF – cfmea*, considers structural and interactive complexity of failure modes and effects it can be used to generate the measure. Based on the I_{cfmea} the selection and evaluation of the component and/or system can be carried out as per failure modes and effects point of view. To evaluate *VPF – cfmea*, numerical values of C_i (effects of component *i*) and a_{ij} (strength of interaction between components) are required**.**

- Reliability index (I_r): The *reliability* index of CBSS design is a quantitative measure of the CBSS constituents. As *reliability* expression, *VPF – r*, considers structural and interactive complexity of CBSS design it can be used to generate the measure. Based on the I_r the selection and evaluation of the CBSS design can be carried out as per reliability point of view. To evaluate *VPF – r*, numerical values of R_i (reliabilities of elements) and r_{ij} (reliabilities of interactions between elements) are required**.**

- *Quality Index* (I_Q)*:* The quality index of a component can be computed by evaluating diagonal elements and establishing relative importance of off diagonal elements of VPQM i.e. *variable permanent quality matrix*. The diagonal elements of the matrix correspond to quality characteristics while off diagonal elements of the matrix correspond to relative importance of one characteristic over other.

Software component classification and quality related contributions

- The SDCS framework is developed that can classify software component on the basis of *architecture level, domain, kind, source, generic functionality* and *phase*. Comparison and evaluation of software component can be performed on a homogeneous set of products such as *server side languages* (*SSL*), *server side*

engines (*SSE*), *client side languages* (*CSL*), *client side engines* (*CSE*) etc. The framework leads to a broader and an in depth classification. This classification is intended to partition software component into sets whose elements are comparable. SDCS framework is useful for several purposes. First, it provides an insight into software component available in the same category, second it depicts the comprehensive information about software component such as its source, domain, functionality etc., and third it gives a purpose to get knowledge, learn, assess, evaluate and compare software component. The framework can be used to denote well known classes of elements, such as classes of all browsers. This framework helps in building sound knowledge and learning process. It also helps in assessing, evaluating and comparing software component.

- The software component quality model is developed. At the highest level the *SCQM* consists of eight characteristics – *functionality, reliability, usability, efficiency, maintainability, portability, reusability* and *traceability*. Comprehensive review of conventional and component specific quality models is done in order to identify their shortcomings. Each characteristics is further reviewed and explored in the form of sub-characteristics and associated attributes respectively. Quality model can be used to evaluate software component and in totality CBSS.

Decision based contributions

- A new approach, decision based concurrent framework, to provide an effective selection of software component for designing CBSS is presented. The approach takes into consideration input from concurrent teams for the selection and ranking of software components. The approach is capable of handling any criteria and any number of alternatives for selecting the optimum one when sufficient resource for the computation is provided. The methods and principles used in the framework – Concurrent Engineering, (Fuzzy)*AHP* and (Fuzzy)*TOPSIS* in both non-fuzzy as well as fuzzy environment are useful in quantifying the software component selection process during the design phase of CBSS. This decision framework as developed helps in identifying suitable software component by considering all the design (criteria) parameters concurrently. A decision Matrix is created on the basis of available

qualitative and quantitative inputs for the criteria/sub-criteria. The relative weights generated from Analytic Hierarchy Process structure act as an input to normalized decision matrix. (Fuzzy) positive and negative benchmarks, i.e., hypothetical (fuzzy) best and (fuzzy) worst software component solutions are generated. The methodology ensures that optimum software component is nearest to the hypothetical (fuzzy) best solution and farthest from hypothetical (fuzzy) worst solution.

9.3 Usefulness of the Developed Methodologies and Frameworks

This section is intended to briefly summarize some high level benefits of the developed approach and methodological framework. Below the roles and the associated benefits are mentioned:

System Analyst: Complete system analysis and evaluation is possible by utilizing system permanent expression. Identification of alternative system design, comparison and evaluation at this stage is facilitated by system's structural identification set.

Quality Engineer: Comprehensive software component specific model is available that helps in exploring quality of a component. Concurrent quality characteristics can be evaluated by utilizing quality matrix and quality function concepts.

Market Watcher: Market place exploration is easier and understandable since market watcher to screen software component according to software component classification framework. Thus software component repository can be easily created.

Knowledge Keeper: Permanent expression when associated with actual terms and given physical meaning can act as a knowledge hub. Each term of the expression with suitable interpretation can be stored properly and later retrieved for further analysis.

Usability Engineer: Component designer and developer can identify potential usability factors affecting component by performing sensitivity analysis in order to improve the overall *usability* of a component.

Maintainability Engineer: Component designer and developer can identify potential *maintainability* factors affecting component by performing sensitivity analysis in order to

improve the overall *maintainability* of a component. Relative *maintainability* indices are helpful in improvising *maintainability* aspects of a component.

Reliability Engineer: Identification of the potential component and/or design failures can be done easily and comprehensively.

Designer: Selection, evaluation and ranking of alternative component and *CBSS* design become easier by utilizing the approaches mentioned in the thesis.

Selector: To take the final decision based on the evaluation of the candidates is more reliable since all the information required is in the same umbrella and therefore their comparison is better handled and are less risky.

Management: Just-in-time, cost effective, stable and good quality product is possible by the involvement of concurrent teams in the project and utilizing methodological frameworks. Management will get total system overview considering all factors and issues related to project and can take appropriate decisions.

9.4 Step-by-step Procedure for the Implementation of Unified Systems Approach

As explained in chapter 1 that a unified systems approach is required to cope up with the modeling, analysis and design of component and component based software systems. This approach must take into account concerns of stakeholders of respected component oriented project/domain. The methodological framework developed in the thesis using graph theoretic systems approach; software component classification framework and concurrent decision based frameworks can be implemented at a software and research industry as is discussed below:

On the basis of the worked carried out in this thesis, a dedicated stand-alone knowledge expert system can be developed to document, compile and evaluate software component and component based software systems. This can be done by utilizing indices such as: *Systems Structure Index, Usability index, Hypothetical best usability index, Hypothetical worst usability index, Maintainability index, Hypothetical best maintainability index, Hypothetical worst maintainability index, Relative maintainability index with hypothetical best maintainability index, Relative maintainability index with hypothetical worst maintainability index, Failure index, Reliability index and Quality Index* and decision techniques such as: *AHP, TOPISIS, Fuzzy AHP and Fuzzy TOPSIS.*

It is to be noted that based on developed approach other quality characteristics indices can also be created, documented and implemented. To get quality system analysis following recommendations and step-by-step procedure is given below:

Step 1: Identification of problem domain, resources and constraints.

Step 2: Development of System Model, see chapter 2

Step 2.1: Identification of sub-systems, sub-sub-systems up to the component level considering all levels of interactions, see section 2.3

Step 2.2: Identification of Quality concerns, followed by sub-characteristics and associated attributes considering all levels of interactive complexity, see chapter 7.

Step 2.3: Identification of software component market place and repositories based on software component classification frameworks and document details accordingly, see chapter 3.

Step 3: Development of matrix model, see section 2.5.

Step 3.1: Developing system permanent matrix and system permanent function, see section 2.5.5

Step 3.2: Repeat step 3.1 up to respective component level.

Step 3.3: Developing quality permanent matrix and quality permanent function, see section 7.4.2

Step 3.4: Repeat step 3.3 up to respective quality attribute level.

Step 4: Performing Evaluation, see chapter 2

Step 4.1: Developing coefficient of similarity and dissimilarity indices on the basis of systems structure, see section 2.8

Step 4.2: Filter software component from repository using concurrent decision based framework, see chapter 3

Step 4.3: Developing indices related to quality and create coefficient of similarity and dissimilarity indices, see chapter 2, chapter 4 to chapter 6.

Step 4.3.1: Develop composite quality index, see chapter 7.

Step 5: Arrange alternatives in ascending order or descending order utilizing results from indices based on step 4.

Step 6: Perform selection and ranking.

Step 6.1: Repeat step 1 to step 5 until satisfied.

Step 6.2 Document results and stop.

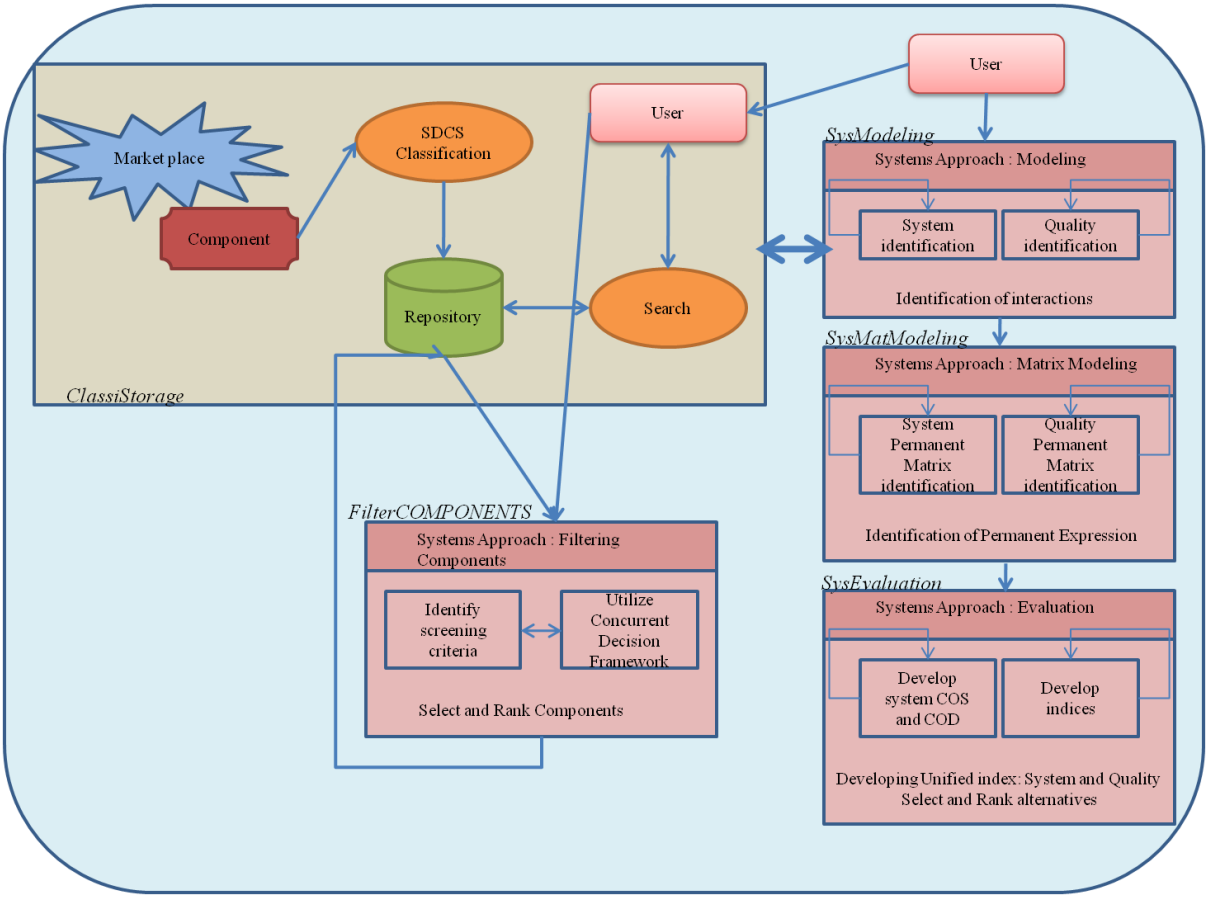

Figure 9.1 Unified systems approach

Above procedure can be followed to create composite quality index based on systems structure and quality characteristics associated with it. Five software sub-systems, Figure 9.1, can be created which allow users to feed values and get required information in order to model, analyze, select, evaluate and rank alternative software component, system designs and strategies.

Sub-system *ClassiStorage*: This sub-system is responsible to generate software component repository based on *SDCS* framework. Effective searching of software component can be done by providing classification keywords. It interacts with software component market place to get updates for updated version of exiting software component and new software component.

Sub-system *FilterCOMPONENTS*: This sub-system screens the initial list of software component fed from sub-system *ClassiStorage* and select and rank software component as per criteria fed into it. Later the final ranking of software component list is stored in *ClassiStorage* repository.

Sub-system *SysModeling*: This sub-system provides the functionality of creating structure of sub-systems and quality characteristics for a particular domain. It also provides the functionality of creating interactions at all levels.

Sub-system *SysMatModeling*: This sub-system helps in generating permanent expressions for systems and quality structure. The data of system and quality structure is fed from *SysModeling* to sub-system *SysMatModeling.*

Sub-system *SysEvaluation*: This sub-system is responsible for evaluating and interpreting terms present in permanent expressions generated by sub-system *SysMatModeling*. Similarity and dissimilarity of system structure and several aforementioned indices can be developed. This sub-system will provide variety of visual aids such as pie-charts, graphs etc., to facilitate decision makers, designers, managers and other stakeholders in making effective strategic decision to select, optimize, evaluate and rank software component and designs.

It is to be noted that some practical concerns such as selection of decision makers, evaluators, designers, developers, integrators etc, may arise during actual implementation of methodology developed in the current thesis. This will affect the elicitation of preference data. The values associated with the attributes and to their interactions need to be determined accurately and precisely to get the actual results. It is to be noted that sometimes these values may be estimations. In case of lack of accurate and reliable data it is recommended to perform multiple runs of the developed model for a what-if or cause and effect analysis.

9.5 Future work

The work presented in this thesis addresses some of the fundamental problems with modeling, analysing and designing component and component based software systems; however, much work remains to be done and several research lines remain open for future investigation to improve and extend the research results obtained from this thesis and it is as follows:

- To explore the use of other techniques to support the evolution and management of the classification schemes thereby providing exhaustive characterization of software component.
- To develop a dedicated web application based on section 9.4 providing XML schema, evaluation tool and structured feedback mechanisms in order to achieve world-wide evaluation
- To utilize the methodological framework developed in the thesis for evaluation of any functional or non-functional requirements.
- To develop capability maturity model for software components that can be mapped to the developed software component quality model.
- To collect empirical industrial data to evaluate and improve the Systems approach, classification framework and decision framework
- To develop hazard analytical model on the lines of concurrent failure modes and effects analysis.
- To exploit developed approach in other software domains of research such as safety-critical domain, banking domain, e-commerce and m-commerce domain etc.
- To extend permanent models with the inclusion of path based approaches.