Study of Component-Based Software Engineering Approach from a Domain Perspective

By

VINAY AVASTHI

Under the Guidance of

DR. DEVENDRA KUMAR PUNIA

(Professor, Head of Department of Information System Management)

&

DR. ASHISH BHARADWAJ

(CIO and Associate Professor, University of Petroleum and Energy Studies)

Submitted

IN PARTIAL FULFILLMENT OF THE REQUIREMENT OF DEGREE OF

DOCTOR OF PHILOSPHY

In

COMPUTER SCIENCE

To UNIVERSITY OF PETROLEUM AND ENERGY STUDIES, DEHRADUN

April, 2012

Dedicated

To

Lord Shiva,

My Parents, Wife and Son

ACKNOWLEDGEMENTS

With deep sense of gratitude, I wish express my thanks and regards to my research guides **Dr. Devendra Kumar Punia**, Professor and Head, Department of Information System Management, University of Petroleum & Energy Studies, Dehradun and **Dr. Ashish Bharadwaj**, Chief Information Officer and Associate Professor, University of Petroleum & Energy Studies, Dehradun for their immense guidance in planning, executing, and writing of my research work in time.

I express my deep sense of gratitude and indebtedness to **Dr. Parag Diwan**, Vice-Chancellor, University of Petroleum & Energy Studies, Dehradun for his continuous support and encouragement during the course of my research.

My thanks are due to **Dr. S.J. Chopra**, Chancellor, University of Petroleum & Energy Studies, Dehradun for his encouragement and providing excellent environment for my research work.

My special thanks to **Ms. Deepa Verma**, Director-HR & IR, University of Petroleum and Energy Studies for her tremendous support, courage and help at each and every step of my work

I am highly grateful to my Dean, **Dr Shrihari**, Associate Dean **Dr. Kamal Bansal,** HOD CIT **Dr. Manish Prateek,** H.O.D Physics **Dr. Piyush Kuchal**, HOD CFC **Prof. Vikram Sahai** for their support, constructive comments and suggestions throughout my work.

This work would not have been possible without the selfless support of **Mr. Vinod Kumar Taneja,** Director Technical, NIC Dehradun. I shall always remain indebted to him for his excellent guidance in this study.

I express my deep sense of gratitude to domain expert **Dr. Vikas Chand Goel**, Scientist F, NIH Roorkee, **Dr. Pradeep Joshi,** HOD Department of Petroleum and Earth Sciences, University of Petroleum & Energy Studies, Dehradun.

I wish to thank my colleagues at CIT and College of Engineering for their support during my research work.

I also express my appreciation to my wife Prabha and son Pranav for their patience and understanding during the course of this research. Specially for my wife I would like to say that you are incredible and everything in my life. Thank you so much, for your care, affection, support, and encouragement to during my thesis work.

I wish to extend my warmest thanks to all those who have helped me directly or indirectly.

Vinay Avasthi

Dehradun

DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

(Vinay Avasthi)

Date: April, 2012

CERTIFICATE

This is to certify that the thesis entitled **"Study of Component-Based Software Engineering Approach from a Domain Perspective"** submitted by **Vinay Avasthi** to University of Petroleum and Energy Studies, Dehradun for the award of the Degree of Doctor of Philosophy (Computer Science) is a bona fide record of the research work carried out by him under our joint supervision and guidance. The content of the thesis, in full or parts have not been submitted to any other Institute or University for the award of any other degree or diploma.

(Dr. Devendra Kumar Punia)

Supervisor

(Dr. Ashish Bharadwaj)

Supervisor

Place: UPES, Dehradun

CONTENTS

Acknowledgements ..... **ii**
Declaration ..... **iv**
Certificate ..... **v**
Contents ..... **vi**
Executive Summary ..... **x**
List of Abbreviations ..... **xiv**
List of Figures ..... **xvi**
List of Tables ..... **xix**
Chapter 1 : Introduction **1**
1.1 Software Industry Overview 1
1.2 Software Engineering 2
1.3 Component-based Software Engineering 4
1.3.1 Introduction to Software Components 5
1.3.2 Approaches for CBSE (In house, OSS, COTS) 5
1.3.3 Benefits of Component-based Software Engineering 6
1.3.3.1 Reduced Development Cost 6
1.3.3.2 Reduced Development Time 6
1.3.3.3 Easy to Manage Software Complexities 6
1.3.3.4 Improved Quality of Software Development Process 6
1.3.3.5 Reduced Defect Density 7
1.4 Domain Engineering 7
1.5 An example of a domain - Hydrology 7
1.6 Software in the identified Domain 8
1.7 Research Motivation 9
1.8 Research Objectives 9
1.9 Research Methodology 9
1.10 Chapter Outline 12
Chapter 2 : Literature Review **14**
2.1 Component-based Software Engineering 14
2.1.1 Evolution of CBSE 14
2.1.2 Types of software Components 16
2.1.2.1 COTS 16
2.1.2.2 OSS 16
2.1.2.3 In house 17

2.1.3 Examples of Software Components 17
2.1.4 Dependency of Components 18
2.1.5 CBSE and Service Oriented Architecture (SOA) 20
2.1.6 Challenges In usage of Software Components 21
2.2 Software Reuse 22
2.3 Software Applications in Hydrology domain 24
Chapter 3 : Analysis of Existing CBSE Models 28
3.1 Software Components Models 28
3.2 CBSE Based Software development Process 31
3.3 Traditional Vs Component-based Software Development Approch 33
3.4 Risks and Challenges in using CBSE 33
3.5 Application Development Life Cycle USING CBSE 35
3.5.1 Sequential Life cycle 35
3.5.2 Chessman Component life cycle 35
3.5.3 "V" Development Life Cycle 36
3.3.4 Analysis 37
3.6 Software Component Model Classifications 38
3.6.1 Interface Specification and Language Support 41
3.6.2 Domain based Classification 43
3.7 Summary 46
Chapter 4 : The State-of-the-art Software Component Technologies 47
4.1 General Purpose Software Component Technologies 47
4.1.1 CORBA Component Architecture 47
4.1.2 Enterprise JavaBeans 49
4.1.3 Fractal 53
4.1.4 Microsoft COM and DCOM 54
4.1.5 Web Services 55
4.2 Domain Specific Component Technologies 56
4.2.1 Automotive Industry 57
4.2.1.1 AUTOSAR 57
4.2.1.2 SaveCCM 58
4.2.2 Robotics 58
4.2.2.1 Fawkes Component Model 59
4.2.3 Embedded System 60
4.2.3.1 Koala 61
4.2.3.2 Pin Component Technology 62
4.2.4 Earth Science 63
4.2.4.1 Common Component Architecture 63

Chapter 5 : Domain Example - Hydrology 66
5.1 Theoretical base of Hydrology domain 66
5.1.1 Hydrology Cycle 66
5.1.2 Hydrology System Concept 68
5.1.3 Application of Hydrology in Engineering and Science 68
5.2 Hydroinformatics : information technology (IT) in domain 69
5.2.1 Flow of Information in Hydrology 69
5.2.2 Data Integration 70
5.2.3 Decision Support Systems 72
5.2.4 Modeling paradigm 74
5.2.4.1 Process Based Modeling 74
5.2.4.2 Data-driven Modeling 75
5.2.4.3 Agent-based Modeling 76
5.2.4.4 Thumb rules of Modeling 76
5.2.5 e-learning in hydrology domain. 77
5.3 Software Use in Domain 79
5.3.1 National Organizations 79
5.3.1.1 National Institute of Hydrology 79
5.3.1.2 Government of India's Hydrology Project II 80
5.3.2 International Organizations 83
5.3.3 Software Companies 83
5.3.3.1 Deltares 83
5.3.3.2 DHI Software's 85
5.4.3.3 Innovyze 85
5.3.3.4 Bentley 86
5.4 Status of Domain-Specific Software in India 88
5.4.1 Organizations 89
5.4.2 Software Applications 91
Chapter 6 : Domain Specific CBSE Framework 95
6.1 Software reusability through CBSE in domain 95
6.1.1 The Rainfall Runoff Software Library 98
6.1.2 Open Modeling Interface 99
6.2 Identification of reusable software components 100
6.3 Need of domain specific Component Based Software Engineering (CBSE) Framework 103
6.4 Model Application 106
6.5 Domain specific CBSE Framework 107
6.5.1 Software Component Development 110
6.5.2 Component based Application development 110

6.5.3 Software component Design Issues 111
6.5.3.1 Atomic design 111
6.5.3.2 Composite design 112
6.6 Data Specifications – When, What, How? 113
6.7 Software Component Repository 114
6.8 Implementation of Software Repository 114
6.9 Component based software development 122
6.10 Summary 129
Chapter 7 : Conclusions and Future Work 132
7.1 Review of efforts and results of the research 133
7.2 Recommendations 144
7.3 Limitations of this research 145
7.4 Suggestions of future work 145

References 147
Annexure A 172
Annexure B 174

EXECUTIVE SUMMARY

There has been a great expansion of software industry in last three to four decades. Software solutions are needed in every domain whether it is automobile, embedded systems, ecommerce, or any other service industry such as banking, telecommunications, or hospitality. Today's advanced hardware technologies, high speed data centers and emerging advancements in web technologies are widely available, yet use of appropriate software development methodologies for successful software systems is a challenge.

Many software development approaches are in use such as Structured, Object Oriented Analysis, Agile, and Component-based Software Engineering. In Component-based Software Engineering approach software applications are developed through reusable software components. This approach reduces development cost, time, complexity, defect density and increases software quality. Software components are a self-contained piece of code with well-defined interfaces which is independent to any application and it can be reused in any other software application based upon specifications. In an organization software components source may be in-house developed or from third parties. Third parties software components are either Commercial Off-The-Shelf (COTS) software components or Open Source Software (OSS) components.

Component-based Software Engineering (CBSE) is an upcoming area in Software Engineering. Its applications in a domain are not yet understood clearly. This research work studies Component-based software engineering approach from a domain perspective i.e. software components as building blocks for domain specific software applications. Hydrology has been identified as an example domain for this research work. Hydrology deals with the science of water on the earth and studies endless cycle of water on the earth. To study the use of Component Based Software engineering approach in domain, extensive literature was surveyed, particularly in the area of

Component based software development, Software reusability and software application in hydrology domain.

This research classified and analyzed 32 software component models in different domains on the basis of Interface specification, language support and whether they are domain-specific or general purpose. Principal findings from this exercise are:

1.) Every software component model has its own interface specifications.

2.) Most of the software component models are being used for general purposes and are not exclusive to a domain.

3.) A limited number of software component models are available for domain specific applications in domains like Earth Modeling, Robotics, Automobiles Industries, High Performance Computing and Embedded systems.

Present research work has addressed some of the identified risks and challenges in using software components by analyzing and studying three Component-based development life cycles, namely Sequential Life cycle, Chessman Component life cycle and "V" development life cycle.

This research compares and presents state-of-art software component technologies including general purpose software component technologies like CORBA, Microsoft COM and DCOM, EJB, FRACTAL and Web Services, and domain specific software component technologies like AUTOSAR and SaveCCM in Automotive industry, Fawkes in Robotics, KOALA and PIN in embedded system, CCA in HPC and Earth science.

Before understanding the applicability of Component-based Software Engineering approach in the identified domain, use of information technology in the domain was explored. In the identified domain of hydrology, information technology is being used in information exchange, data integration, decision support system, e-learning and modeling paradigms like

process based modeling, data-driven modeling and agent based modeling. Main Sources of software in hydrology domain are national organizations, international organizations and software companies like Deltares, DHI software, Innovyze and Bentley, etc.

A survey was carried out to determine the software being used in the domain. It was found that software in hydrology domain in India is being used by State, Central Government Organizations and Academics & Research institutes. Another finding of the survey was that there was very little awareness about the latest developments and best practices in the area of software application development, for example software reusability through OOA, CBSE and Software Product Lines, etc. Barring a few cases in some of the research institutes (15%-20%), where concept of reusability was being used but not through technology like software components.

One of the application areas from the domain, surface water was identified for further study. 40 software were studied in detail to find out the use of software reusability using Component-based Software Engineering approach by using an experimental setup, each with a computer machine and operating system confirming to the requirements of the software-to-be-analyzed. The outcome of our experimental setup was that there was a very limited role of software reusability through Component-based software engineering in hydrology's surface water domain-specific software under consideration. Partial use of software reusability was however found.

To understand how can software components be used/ reused in domain, this research work first justify the need of domain specific Component-based Software Engineering framework and then has proposed a domain specific Component-based Software Engineering framework that includes the development of software components as well as development of application from these reusable software components.

A web-based software components repository was developed that contained domain specific software components, general purpose software components, Open Source Software (OSS) components and Commercially off-the-shelf (COTS) for the use of water scientists, engineers, hydrologists, researchers and water managers to refer to, use and reuse them whenever there is a need of designing, developing, augmenting or modifying domain-specific software applications. A software application was developed using above software component repository to demonstrate the applicability of software components and the framework.

Further development in software components could be in the direction to design and development of Component as a Service (CaaS) on cloud platform.

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
API	Application Programming Interface
CaaS	Component as a Service
CAD	Computer Aided Design
CBRS	Component Based Robotics Application
CBSE	Component-based Software Engineering
CCA	Common Component Architecture
CCL	Construction Component Language
CDL	Component Definition Language
COM	Common Object Model
COTS	Commercial Off-The-Shelf software / components
CUAHSI	The Consortium of Universities for the Advancement of Hydrologic Science Inc.
DCOM	Distributed Component Object Model
DHI	Danish Hydraulic Institute
DSS –P	Decision Support System – Planning
EDI	Electronic Data Exchange
EJB	Enterprise Java Bean
GIS	Geographical Information System
HP Labs	Hewlett-Packard Laboratories
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICT	Information and Communications Technology
IDL	Interface Definition Language
IEEE	Institute of Electrical and Electronics Engineers, Inc.
IMD	Indian Meteorological Department (GoI).
ISO	International Standards Organization
J2EE	Java 2 Enterprise Edition

JMS	Java Message Service
KM	Knowledge Management
NIH	National Institute of Hydrology
NSF	National Science Foundation
OMG	Object Management Group
OMS	Object Modeling System
OOA	Object Oriented Analysis
OpenMI	Open Modeling Interface
ORB	Object Request Broker
OSGi	Open Services Gateway initiative
OSS	Open Source Software
PRMS	Precipitation Runoff Modeling System
RMI	Remote Method Invocation
SaaS	Software as a Service
SCA	Service Component Architecture
SCDL	Service Component Description Language
SDLC	Software Development Life Cycle
SIE	Software Engineering Institute
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
UDDI	Universal Description Discovery and Integration
UML	Unified Modeling Language
WSDL	Web Services Description Language

LIST OF FIGURES

Figure No.	Title	Page No.
Figure 1:	Component-based Software Engineering Process	4
Figure 2:	Research Approach	11
Figure 3:	The Evolution of Software Components.	15
Figure 4:	Software Component view of OIMS	18
Figure 5:	Framework of COM Based System Dependency Analysis	19
Figure 6:	Software Component Dependency Graph	19
Figure 7:	Service Component Description Language Example	20
Figure 8:	River Width Calculation Output of RIVWIDTH Tool	24
Figure 9:	Calculation for load uncertainty through DUET-H/WQ	26
Figure 10:	Software Components	29
Figure 11:	Component Diagram	30
Figure 12:	Basic elements of Software Components Model	31
Figure 13:	CBSE Based Development Process	32
Figure 14:	Sequential Approach Software development through CBSE	35
Figure 15:	Component Development Life Cycle.	36
Figure 16:	"V" CBSE based Software development life cycle	36
Figure 17:	Kue Lau Categories based on composition	39
Figure 18:	Components Domain	45
Figure 19:	Domain wise % of component models	46
Figure 20:	Role of ORB in CORBA	48
Figure 21:	EJB Container	49
Figure 22:	Type of Beans in EJB	50
Figure 23:	Clients interacting with an EJB component system	52
Figure 24:	A Fractal Software Component	53
Figure 25:	Microsoft component layers	55
Figure 26:	Web Service	56
Figure 27:	Composition of Web services	56

Figure 28: AUTOSAR system architecture 57
Figure 29: Fawkes blackboard 59
Figure 30: Component Composition - Koala 61
Figure 31: Structure of the Pin Component Technology 62
Figure 32: CCA Component Architecture 64
Figure 33: Integration through ports in CCA 65
Figure 34: Hydrology Cycle 67
Figure 35: Hydrology System Concepts 68
Figure 36: Flow of Information in Hydrology System 69
Figure 37: HydroServer Architecture 71
Figure 38: Commercial Software used in HydroServer 71
Figure 39: Thumb Rules of Modeling. 76
Figure 40: Features of DSS-P 81
Figure 41: Software Used in DSS-P 81
Figure 42: Data Processing & Analysis in HIS 82
Figure 43: Open communication between Delft-FEWS and models 84
Figure 44: Applications in FloodWorks 86
Figure 45: Flood Works Activities 86
Figure 46: Bentley's Software 87
Figure 47: WaterGEM Contribution in Revenue Performance 88
Figure 48: Hydrology domain Software user 90
Figure 49: Software Usage 90
Figure 50: Software categorization in hydrology domain 96
Figure 51: Parameters for Software analysis 96
Figure 52: Model Selection in RRL Library 98
Figure 53: Input Output Exchange within Model 99
Figure 54: Application Wise Categorization of Surface Water Software 100
Figure 55: Software Application Area 101
Figure 56: Model Application in Hydrology Domain 107
Figure 57: Domain Specific CBSE Framework 109

Figure 58: Atomic Components 111
Figure 59: Composite components 112
Figure 61: Bidirectional composition 112
Figure 60: Unidirectional Composition 112
Figure 62: Datasets value 113
Figure 63: Login page of Software Component Repository 115
Figure 64: Domain specific software component repository 116
Figure 65: Software Components in Repository 117
Figure 66: Software component information – Hydrograph 118
Figure 67 : Software component information – Rain Fall Runoff 119
Figure 68: Different Buttons in Software Repository 120
Figure 69: Supporting feature in the repository 121
Figure 70: Flow chart to calculate infiltration and runoff 125
Figure 71: User interface 127
Figure 72: Time Series data import to model 127
Figure 73: Hydrograph Corresponding data Set II 128
Figure 74 : Hydrograph Corresponding data Set II 129
Figure 75 : Classification of Software Components 136
Figure 76: Software Usage Areas 139

LIST OF TABLES

Table No.	Title	Page No.

Table 1: Analysis of Traditional and CBSE Approach 33
Table 2: Summary of CBSE Risk and Challenges 34
Table 3: Risk and Challenges addressed by CBSE Lifecycle Process 37
Table 5: Interfaces Languages 42
Table 4: Interface Type 43
Table 6: Classification of Component Model Domain Specific / General 43
Table 7: EJB 3.0 Specification 53
Table 8: Hydrological Components 67
Table 9: Software from NIH 80
Table 10: Software used in hydrology domain 91
Table 11: List In House Developed Software Applications 94
Table 12: Sample Time Series Data 124
Table 13:Equations for Rainfall Runoff model 126
Table 14: Interface Based Classification 134
Table 15: Language Support 135
Table 16: Summary of Major findings 142

CHAPTER 1

INTRODUCTION

This chapter introduces the topic and presents an overall overview of the software industry globally and in India.

1.1 SOFTWARE INDUSTRY OVERVIEW

In last three to four decades, there has been a great expansion of Software Industry. Software solutions are needed in every domain whether it is automobile, embedded systems, ecommerce, or any other service industry such as banking, telecommunications, or hospitality. Main driving forces in software development process are cost, meeting deadlines and quality. The main constituents for the cost are skilled technical manpower, and rapid changes in the technology. Scheduling is another important force for software projects. Software need to be developed faster and project should be completed within stipulated time. Unfortunately, history of software development shows there are lot of delays in meeting deadlines (Yu, 2009). Beside cost and schedule, today, software quality is also main mantra for developing better software products.

Many software companies have reported problems related to software development process, as a 'software crisis' occurred in last few decades. So far, software industry has achieved many milestones, still it need improvement in software development process as Robert Glass states *"I look at my failure stories*

and see exception reporting, spectacular failures in the midst of many successes, a cup that is [now] nearly full" (Glass,1998).

One of the key challenges in software development process is to understand the core domain knowledge and to align this domain logic with current software development technologies and practices (Hidalga et al, 2008).

Efforts of individual for improving software development process can never be neglect even if high speed data center or advanced hardware technologies are being used (Brian, 2012).

Open source and Software as a Service are the new gateways to satisfy user need in the software industry, even through open source needs more expert man power.

1.2 SOFTWARE ENGINEERING

Software engineering is collection of technologies that apply with engineering approach to development and maintenance of software applications. Here technology means principle, development approach, tools, software process etc. In standard IEEE glossary software engineering definition given as

> *"Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software; that is, the application of engineering to software"* (IEEE, 1990).

Software engineering is different from Civil, Electrical and Mechanical engineering in this regard, Philippe Kruchten suggests differentiating characteristics of software engineering as follows:-

- Lack of fundamental theories makes Software Engineering sometimes difficult to learn about software development process without developing it.

- Changes in software are encouraged, but the impact of these changes is difficult to predict.
- Assessment of new technologies is not properly done due to rapid evolution of technologies.
- Software development cost does not include bug fixing, updates and re-design (Basili et al., 2008).

Current challenges in software engineering are new software development processes, Open Source Software (OSS) , Commercially Off-The-Shelf (COTS) and more systematic synthesis and collection of worldwide domains (Basili et al., 2008).

There are many software development approaches such as Structured, Object Oriented Analysis (OOA), Agile, Component-Based Software Engineering (CBSE).

Structured programming is based upon top-down design model. In this approach entire program is divided into modules and sub modules and it is programmed accordingly. Structured programming provides efficient way to load the program in to memory and modules can be reused in same or other programs. (Darcy et al., 2005).Two mathematicians Corrado Bohm and Guiseppe Jacopini demonstrated the concept of structure programming through only three constructs: sequences, decisions, and loops.

Real world problems can be solved effectively through Object Oriented approach. Object oriented approach provides mechanism to support reuse of coding block, design and analysis. In OO approach design phase consists of four different steps: system design, class design, interface design and database design (Booch, 1996). In OO approach, classes and objects are the main constructs from analysis to implementation phase. Object-oriented languages such as C++, Java provide the feature of encapsulation, inheritance, data binding, encapsulation and polymorphism also.

Agile software development approach is focus on incremental specification, design issuse and implementation with iterative development approach and provide full integration testing and development (Sommerville,2010; Talby et al., 2006). eXtreme Programming(XP) is one of the prominent way for Agile software development (Auer and Miller, 2001).

1.3 COMPONENT-BASED SOFTWARE ENGINEERING

Component-based Software Engineering (CBSE) become known in late 1990s. Motivation behind the development of this approach is, software developer are not satisfied by reusability approach through object oriented analysis. It is the process of defining, selection, implementation, retrieving, integrating loosely coupled software components into applications.

In CBSE, software applications are build from pre-developed software components, it enhances the degree of software reusability, and the software applications are developed with reusable software components through integration mechanism like interface, pipe, glue code. Figure 1 depict the principal activities of CBSE process by Sommerville (2010).

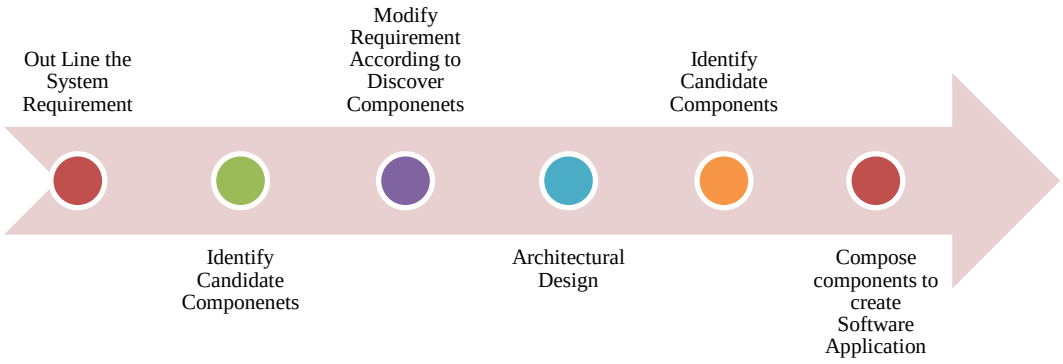


Figure 1: Component-based Software Engineering Process

1.3.1 INTRODUCTION TO SOFTWARE COMPONENTS

In recent years, number of definitions for "software component" has been proposed. Szyperski provided a very popular definition of software components (Szyperski, 2002)

> *"A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties."*

The Object Management Group (OMG) also provides another similar definition where group of objects are also treated as software Components (OMG, 2003)

> *"A component represents a modular, deployable and replaceable part of a system that encapsulates implementation and exposes a set of interfaces."*

In summary, software component is a self -contained piece of code with well-defined interface which is independent to any application and it can be reused in any other software application also. Software components can be placed on any network node, depending upon the needs of particular application domains and despite of the type of particular network structure. While developing software component, one should keep in the mind about the reusability aspect of software components regardless the future need of organization.

1.3.2 APPROACHES FOR CBSE (IN HOUSE, OSS, COTS)

In Component-based Software Engineering, software development can be performed using existing software components from in-house or from third parties. Third parties software components are either Commercial Off-The-Shelf (COTS) software components or Open Source Software (OSS) components. COTS are commercially available and are generally a black box in nature, only a

few COTS are available with source code. Whereas, OSS components come with source code and more or less freely available (Li et al., 2004). COTS software components are generally available with previous three releases and support to earlier versions is also provided by third parties (Basili and Boehm, 2001).

1.3.3 BENEFITS OF COMPONENT-BASED SOFTWARE ENGINEERING

1.3.3.1 REDUCED DEVELOPMENT COST

Component-based Software Engineering reduces the cost of software development by using the concepts of software reusability. It means there is no need to write code from the scratch. This saves resources and indirectly saves the cost.

1.3.3.2 REDUCED DEVELOPMENT TIME

CBSE reduces the development time because all software components are pre tested and debugged properly. So when software components are used, it requires less time for debugging and testing the application.

1.3.3.3 EASY TO MANAGE SOFTWARE COMPLEXITIES

Through component-based software engineering, the complexities of software development can be easily managed because all software components are pre tested and verified and are properly documented. Any anomalies can be easily identified and rectified.

1.3.3.4 IMPROVED QUALITY OF SOFTWARE DEVELOPMENT PROCESS

Software components are often repeatedly reused, and in the process, almost all types of errors can be detected and corrected. This ensures high quality of software development process.

1.3.3.5 REDUCED DEFECT DENSITY

Defect density is calculated by dividing no of defects by no lines in the code. As compared to applications designed without using reusable software components, applications developed through CBSE have less defect density because software components are pre-tested and configured.

1.4 DOMAIN ENGINEERING

Czarnecki proposed that *"domain engineering is the activity of collecting, organizing, and storing past experience in building systems or parts of systems in a particular domain in the form of reusable assets (i.e. reusable work products), as well as providing an adequate means for reusing these assets (e.g. retrieval, qualification, dissemination, adaptation, assembly) when building new systems"* (Czarnecki and Eisenecker, 2000). Major steps in domain engineering are 1). Domain Analysis, 2). Domain Design 3). Domain Implementation (Hoffer et al.,2005).

1.5 AN EXAMPLE OF A DOMAIN - HYDROLOGY

For this research work, Hydrology has been identified as an example domain. Hydrology deals with the science of water on the earth and study of endless cycle of water on the earth and its environment is known as hydrology cycle. Knowledge of hydrology is applied to use and control of water resources for the benefits of society. Water resources on earth are affected by changes in distribution channels, temperature of earth and circulation of water on the earth. Water resources are also affected by following human activities:

- Pumping ground water
- Building dams
- Deforestation
- Use of excessive fertilizers

- Irrigation of land

Beside these there any many constructive and destructive activities that affect the quality and circulation of water on the earth. Data related to Hydrology is very important and Hydrology data used in following areas:

- Flood Control
- Hydro power generation
- Irrigation
- Water Distribution urban and rural area
- Pilgrimages (Badhrinath, Kedarnath, Yamunotri, Gangotri, Haridwar)
- Snow Gagging, / Glacier
- Dam Water Management
- Ecology
- Industry demand

Hydrological data used by scientists/researchers, agriculturists, planners, administrators/managers, engineers, forest, municipal corporations and army officials to achieve the aforementioned welfare activities.

1.6 SOFTWARE IN THE IDENTIFIED DOMAIN

Software's in hydrology domain are used by hydrologist, water mangers, researchers, scientists and academicians. Main functional areas of software in this domain is Data integration, GIS, Scenarios, Optimization, Analyses, Reporting, Studies and Workflows management. Most of the software in this domain is used for implementations of models, integrations with GIS, CAD, 2D and 3D graphics and implementation of physical laws in terms of mathematical algorithms. Commercials software's in hydrology domain are provided by many vendors like Deltares, DHI Software's, Innovyze, Bentley etc. At present, software usage is

very limited in the hydrology domain. There is a need to explore the feasibility of developing more software solutions with using software reusability.

1.7 RESEARCH MOTIVATION

Key challenges in software development are complexity, dependency on domain engineering, delays in delivery resulting in overshooting budgets (Sommerville, 2010). Newer approaches like CBSE need to be studied for their applications to various domains. There is a need for software solutions in hydrology domains to integrate various data sources (remote sensing, ground measurements, surface water analysis etc.), various types of models, work flow management and decision support processes. Any study to improve software process for the domain would be greatly beneficial to the various stakeholders like Hydrologist, Water Managers, Scientist, Research Scholars etc.

1.8 RESEARCH OBJECTIVES

- To analyze and compare Software Components Models used in Component Based Software Engineering (CBSE).
- To discover the role of Component-Based Software Engineering in the software used in domain (Hydrology domain as an example).
- To design a framework for software components and framework for Component-Based Software Engineering approach for a domain problem.

1.9 RESEARCH METHODOLOGY

In science and engineering, there are well defined research methodologies. For example, double blind studies of medicines and the experimental setup for physics, chemistry, and engineering discipline with some guideline to perform an experiment with well defined scope of work (Gieryan, 1999).

Software engineering lack such well structured research methodologies and there is a shortage of well define research guideline (Shaw, 2003). Research in software engineering mostly put forwards emerging technologies, rapid development and emerging technologies. There is lack of empirical studies with proper validation (Glass et al, 2004). The research in software engineering should aim to explore one or several parameters to describe reasoning about specific applications and methods to be opted (Endres and Rombach, 2003). Following research methodologies are commonly used in software engineering:

Case Study - It is based upon the investigation of the effects of one or more activities or methods on development process or technology over a given period of time (Yin, 2003) for example defect analysis in .NET and Java based applications. Main goal of Case study is to link, established and trace the particular attributes.

Experiment - In software engineering research experimental setup is used only in university research, where in industry this approach is not so common. Major findings of the study conducted by Sjøberg on type of research in software engineering in the twelve leading software engineering journals have only 1.9% research articles belonging to research in software engineering through experiments (Sjøberg et al., 2005). In another study, less than 2% research studies were found to be done using experimental setup (Ramesh et al., 2004).

Action Research- Proactively involved researcher in the operations and change during the investigation still continue to perform operations. In this scenario, research and practices are allowed to be informed and investigates regularly throughout research (Davison et al., 2004).

Survey - In survey data is collected from sample of population through questionnaires, email, personal interaction or through phone, which is further used to describe effects and correlation (Robson, 2002; Conradi et al., 2005).

Grounded Theories - In this methodology, on the basis of empirical data general or abstract theory is developed. Emerging categories are created through data comparison using grounded theory. The commonly used technique in grounded theories is textual analysis (Creswell, 2003).

In every research, selection of an appropriate methodology fitting to research objectives is very important. Research approach adopted in this research is outlined in figure 2, which is adopted from Shaw's work on how to do good software engineering research (Shaw, 2003).

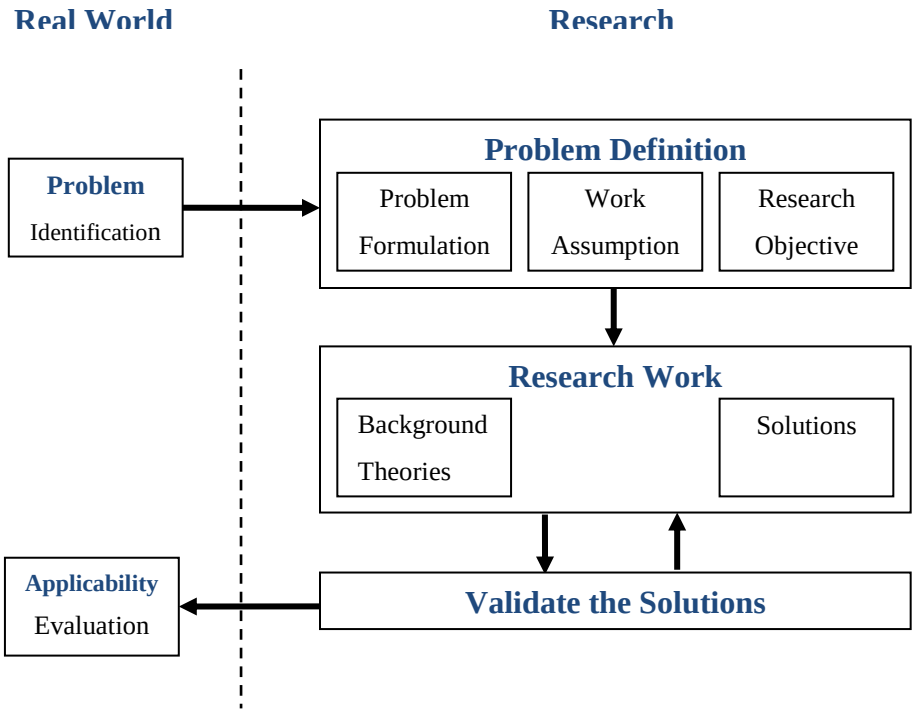


Figure 2: Research Approach

This research focuses on understanding the present scenario of Software Component Models in Component Based Software Engineering along with the role of CBSE in a particular domain. This is descriptive research which uses the

document analysis as the research methodology and secondary data analysis as the data collection method. The last research objective is to design a framework for software components and CBSE approach for a problem in a domain. To accomplish this objective, existing frameworks were reviewed and a new framework was developed using document analysis. Software simulation was used to test the framework after writing code in open source. A repository of software components for the hydrology domain was also developed in open source to demonstrate the application of the framework.

For first research objective, 32 state-of-art software component models were analyzed and compared on domain specificity, interface specifications.

For research objective two, data was collected from different sources through visits, email or telephonic interviews.

For research objective three, possibility of reusable software components in domain was explored based on the data collected for first two objectives and experimental setup. A Domain-specific CBSE framework was developed and implemented as well with OSS.

1.10 CHAPTER OUTLINE

The thesis is organized in seven chapters as discussed here. Chapter 1 introduces the problem, gives brief literature review and discusses the research methodology related to the work.

Chapter 2, 'Literature Review' deals with the extensive literature review in the area of component based software engineering, software reuse, and software application in hydrology domain.

Chapter 3, 'Analysis of Existing CBSE Models' addressed some of the identified risks and challenges of CBSE approach by analyzing and studying

three Component-based development life cycles and presents major findings of comparison and analysis of various software component models.

Chapter 4, 'The State-of-the-art Software Component Technologies' provides overview of existing software component technologies. General purpose as well as domain specific software component technologies.

Chapter 5, 'Domain Example: Hydrology' gives an idea of ICT and software use in hydrology domain. Software use data was collected from State & Center Government organizations and Academic Institutions. Detailed analysis of software in hydrology domain is presented in this chapter.

Chapter 6, 'Domain Specific CBSE Framework' explores the possibility of reusability of software components in domain, which helps in design of the Domain Specific Component-based Software Engineering Framework. This Framework is supported by Technical case through implementation.

Chapter 7, '**Conclusions and future work'** Present review of efforts and results of the research , final conclusions drawn from the study. Recommendations have been made for further research work in this area keeping in view, the potential practical application of this result in engineering and industry.

CHAPTER 2

LITERATURE REVIEW

This chapter presents the review of related literature on component based software engineering, software reuse and software applications in hydrology.

2.1 COMPONENT-BASED SOFTWARE ENGINEERING

Component-based software engineering (CBSE) provides an electrifying and talented standard for developing software. This is often proposed as a solution to the problem of 'software crisis' mentioned by many researchers in the field of software engineering.

2.1.1 EVOLUTION OF CBSE

The Component-based software engineering (CBSE) approach is in forefront of new approach to develop large and complex software applications. It has matured over the last 15 years and with rapid merging of internet based technologies has lead to CBSE as one of the main approach of software development.

Software outsourcing has become very important in software industry. The state-of-art practices of CBSE are analyzed in Germany, Italy, Norway during 2004-2005. In the same context a quality software engineering survey on software industry of china emphasized that translation of software engineering terms from English to Chinese leads to many misunderstanding and confusion. The software professionals are not aware of the best software engineering practices. there is a

need of standard guidelines to be developed using recent software development practices like CBSE (Junzhong et al., 2008).

Figure 3 shows the evolution of software development process by peter(1999) including introduction of software components in late nineties.

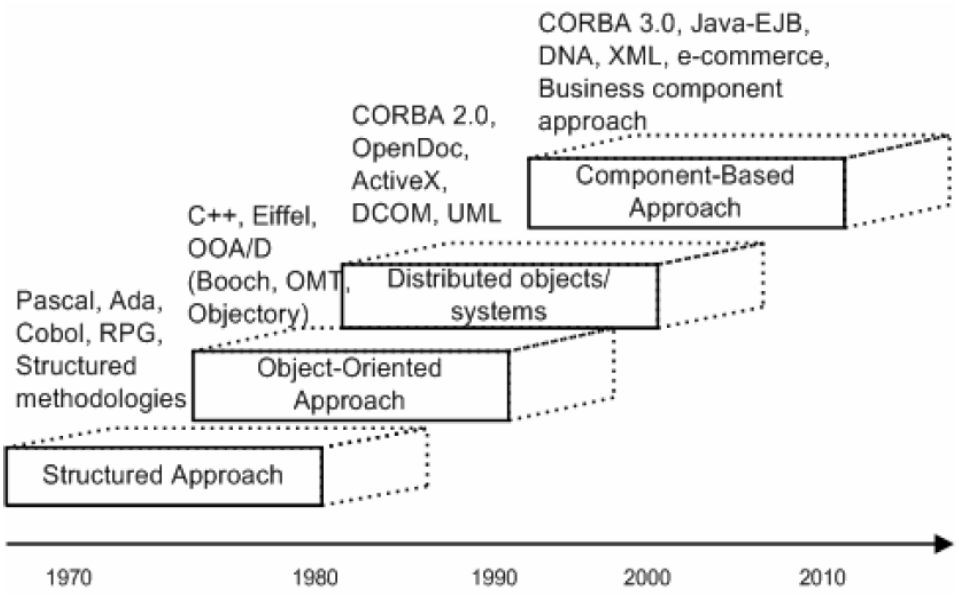


Figure 3: The Evolution of Software Components.

Software components are the key parameters in CBSE. Software professional community working continuously towards to adopted new methods to increased quality and productivity of development process of software components. The traditional software development process is neither fit nor sufficient to develop complex software components. To develop software components within budget, time frame software components metrics are used for efforts estimation, scheduling accuracy, cost analysis and understanding of software development process (Goswami et al., 2009; Sedigh-Ali et al., 2001).

2.1.2 TYPES OF SOFTWARE COMPONENTS

Software components may OSS, COTS or in house developed. CBSE is more focused upon re-use rather then re-inventing the software solutions (Szyperski,2002). There are six main dimension for reuse of software code like substance, Scope, Approach, Techniques, usage and result product (Vliet:, 2008). All these dimension are applicable to software components either it is OSS, COTS or in house developed (Li et al, 2004).

2.1.2.1.COTS

Many researcher, developer work on Commercial Off-The-Shelf software components (Oberndorf, 1997). COTS are ready to use software code that are commercially available for use. Only few organizations providing COTS like component factory, Component Source (ComponentFactory,2012; ComponentSource,2012). Main characteristics of COTS component are they are black box in nature, and documentation, evolution and version controlled are handle by respective vendor (Basili and Boehm, 2001). Usage of COTS leads to decline cost, development time and improved the quality and productivity of software applications (Voas, 1998b). However there are risk also associated with COTS (Voas, 1998a).

2.1.2.2. OSS

Evolution and availability Open Source Software components depend upon contribution provided by OSS community (OSI, 1998-2012). I am using following definition of OSS:

Open Source Software is software being developed under a license compatible with FSF or OSI licenses (OSI, 1998-2012 ; FSI, 1985-2012).

OSS are a little different then freely available software (GNU, 2005). OSS are available in public domain for faster development, OSS component are efficient, reliable and robust in nature. OSS prevent our software application from instability of vendor support (Ruffin and Ebert,2004).The parallel contribution by many developer through Open source communities can fix the bug quickly. Additional license not required for additional usage (Madanmohan and De, 2004). Software companies are widely adopted the OSS components during software development process (Hauge and Sørensen et al, 2009).

2.1.2.3 IN HOUSE

Many Organizations developed their own software components that can be reused in many in house applications. All leading software campiness maintaining the repository of in house developed software components for the future reuse. In house developed software components are more reliable trustworthy and free from proprietary issues.

2.1.3 EXAMPLES OF SOFTWARE COMPONENTS

ABB group, main player in electrical engineering and technology, provides solution in the areas of power generation, transmission, distribution, industrial automation process etc. They applied concept of software reusability through CBSE in their software solution for industrial process control system. Most of the software components developed for this system are in Unix platform and Microsoft Windows NT platform, which are very successful system designed through CBSE approach where emphasis is given on software reusability through software components.

Many challenges are also faced during the adoption of CBSE e.g. software component compatibility with other software components due to changes in technology, increased cost of project due to redesign of software components etc (Crnkovic and Larsson, 2000b).

Advant Open Control System (OCS) designed an Information Management Station for oil producing process in which they transmitted the data to headquarter via a software component through satellite and integrate other modules also through this component as shown in figure 4. In entire system lots of software components are used. Advant's OCS architecture can easily be used to develop and design applications through CBSE approach (Advent, 2011).

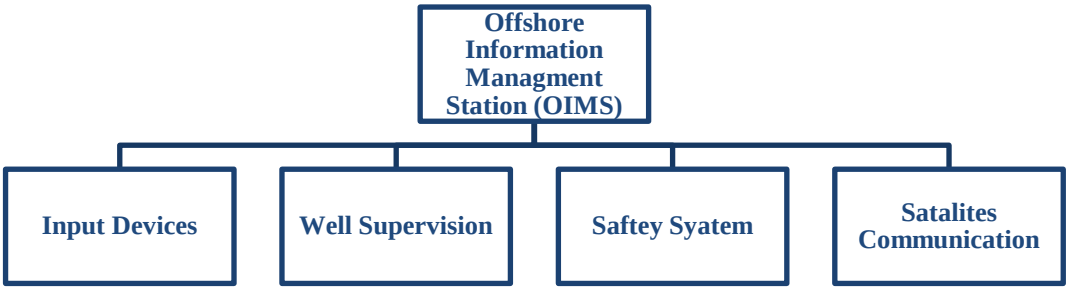


Figure 4 : Software Component view of OIMS

2.1.4 DEPENDENCY OF COMPONENTS

The potential of one component affect other components during the software development process is dependency of software components. Software component dependency is another area of research in CBSE. Understanding of dependency helps in solving many issues that occurs during software development through CBSE approach like integration, testing, software component reuse and component selection. .NET software components are self describing through assembly, metadata about source code and interfaces is available in assembly. EJB components dependencies can be analyzed through

Aspect Oriented programming. Binbin (2010) developed a framework for COM component dependency as shown in figure 5.

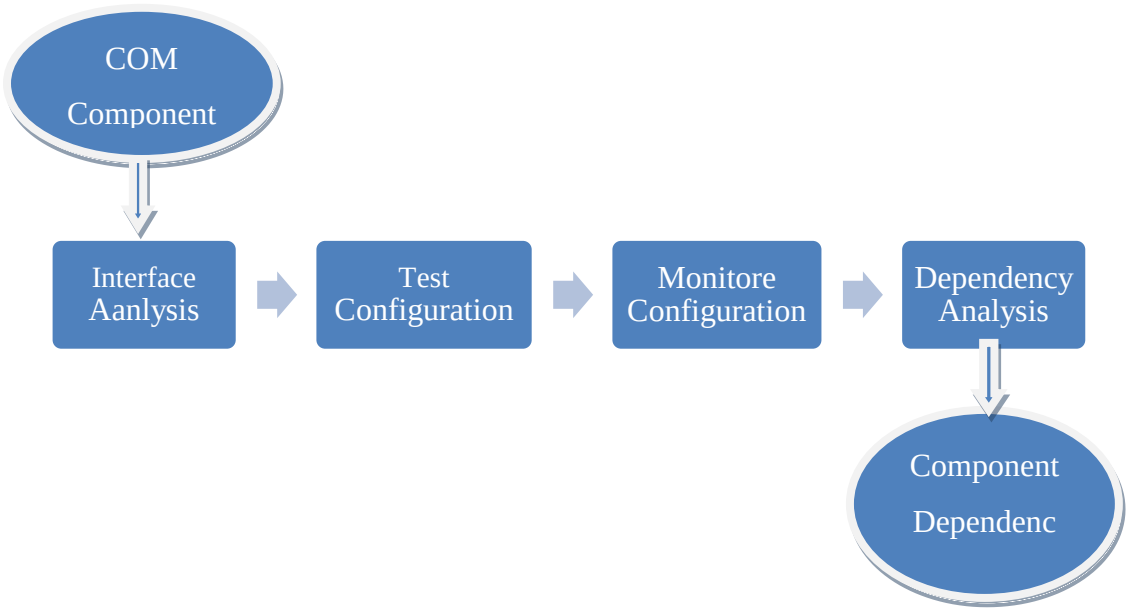


Figure 5 : Framework of COM Based System Dependency Analysis

This dependency analysis framework of COM components has four modules and takes input a COM components and the output is a Component dependency graph. For example in four software components namely W, X, Y, and Z, where dependency are like this . Represented by dependency graph shown in figure 6.

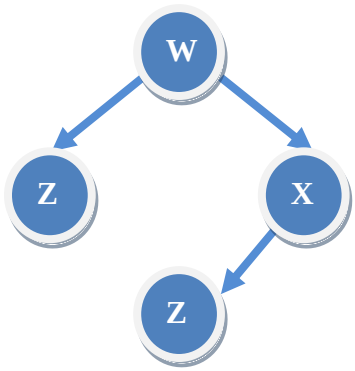


Figure 6 : Software Component Dependency Graph

Dependency analysis are very beneficial, especially when source code of software components is not available and licensing constraints do not allow modification in software components (Binbin et al. 2010).

2.1.5 CBSE AND SERVICE ORIENTED ARCHITECTURE (SOA)

Service-Oriented Architecture (SOA) is advance paradigm of software component reuse through web services. One of the way to implement SOA paradigm is environment of J2EE technologies. Java Business Integration and Service Component Architecture is taken as first step to develop SOA (Vescan et al. 2008) on server side. Service composition is specification of Service Component Architecture (SCA). For this XML based Service Component Description language (SCDL) is used. SCA has minimum description about Components through SCDL depicts in figure 7.

```
   <composite
   <component
      name="AccountServiceComponent">
     <service name="AccountService">
     <interface.wsdl
   interface="http://Account#wsdl.interfac
e(Account)" />
      <binding.ws
uri="http://localhost:8085/AccountService"
/>
       </service>
       <implementation.java
         class="account.AccountImpl" />
   </component>
   </composite>


   <component name="AccountClient">
    <implementation.java
      class="account.AccountClient"/>
      <reference name="AccountService">
   <binding.ws
wsdlElement="http://Account#wsdl.port(Acco
untService/AccountSoapPort)"/>
      </reference>
   </component>
   </composite>
```

Figure 7 : Service Component Description Language Example

SOA principle restricts the interaction up to interface only. Software components may be implemented in any of the technologies like Java, C++, Spring and also supports all type of communication protocols like SOAP, JMS, HTTP, RMI etc. This solution is based upon specification using OSGi, SCA and OMG (Ruiz et al. 2008).

OSGi is used to deploy java components along with native libraries or software components developed in language like Python, Ruby and PHP etc. SCA domain has TargetManager and NodeManager components and TragetManager is actually responsible for carrying out the deployment process within the domain.

There are number of areas in large and complex software development process where cooperation is required. These types of cooperation have their own advantages as well as disadvantages. Great research need is identified in collaborative software development environment. In this context CBSE and SOA can play a very important role (Xinyu et al. 2009).

2.1.6 CHALLENGES IN USAGE OF SOFTWARE COMPONENTS

In CBSE Approach, components selection is the biggest challenge. Lots of research have been done on component selection techniques but more work is required in this area (Vescan et al. 2008).

Software component selection is a complex bustle that needs advanced algorithms, techniques and specialized tools. When COTS software components are used, many risk are involved like supplier risk, obsolescence risk and performance risk. Limited tools are available to support software development process through black box software components. (Hutchinson and Kotonya., 2006).

Component-based Software Engineering works with OSS, COTS and in house developed components. Development of the software is an intellectual process.

Commercially off-the-shelf (COTS) components are black box in nature. Main techniques for software development through COTs are 'The Analytic Hierarchy Process', 'Simple Multiple Attribute Rating Techniques', 'Utility Theory', and 'Weight Score Method' is discussed by Hutchinson. Major challenges in COTS components related to support for software components by vendor and quality concerns (Voas, 1998a).

Kukko also discussed the knowledge management challenges in reuse of software through Component-based software engineering. Technical cases have been taken from large software development companies. Many changes of KM are identified during renewal process of software. In CBSE KM play an important role in earlier design and requirement phase. People should have right attitude to sharing the knowledge. So that KM can be used in Cutting-edge software development technologies (Kukko et al. 2008).

2.2 SOFTWARE REUSE

In this section I have discuss, impact of software reuse on business and its major challenges to industry and academia. Applicability of software reusability is in science and engineering. Software reusability will increase software product quality, services, and profits as well. Software reusability will only succeed if it makes good business sense. Organizations invest capital on software reuse only when there is a possibility of profits to shareholders. Important challenges to support reuse programs are: technology transfer, process focus, organizational issues, corporate strategy and reuse (Favaro et al. 1998).

Software reuse seems to be interesting, however in reality source code sharing among the users is a great challenge. Three cases are presented by Grinter. He concern about code salvage, availability of source code to others have their own issues and challenges and The final challenge for software reuse is re-contextualization (Rebecca and Grinter. 2001).

Software reuse tightly coupled with programming languages in following ways: first, programming languages are evolved from zeroes and one to assembly, subroutines, then modules, classes, components etc. and secondly off the shelf libraries are available in C#, Java, Visual Basic and Visual C++, thirdly the World Wide Web gives the defacto standard library of reusable software assets.

Considerable progress has been made on software reuse, still some issues are left such as scalability, design of large reusable software components, sustainability of reuse, reliability and safety.

Safety is a major challenge specifically for embedded system in the domains like aerospace and automobile. In these domains new architecture is introduced to motivate the use or reusable software components. These two domains are safety critical domain, so during development of software application through reusable software components, Safety issues should be taken care. Standard regulation are introduces for reusing software components in these type of domains (Fraks et al. 2011)**.**

In engineering and science, software reuse is treated as an empirical discipline. Reuse and reusability of software lead to fundamental questions like how to measure reusability and its impact to productivity and quality.

Software metrics are defined and used to describe software reusability. Software metrics are quantifiable dimension of the attributes of a software product. Software metrics are used in many areas like amount of reuse, reuse of libraries, cost, complexities, etc. There have been a few correlation studies of relationship between reuse, productivity, quality and a few in the field of study of methods of reusable components. Such studies deal with typical problems of field studies and try to control the internal validity through the experiments. Measurement and experiment on reuse of software is a area where further work is needed to be done (Frakesand and Terry, 1996).

Software companies are involved in research in software reusability like in HP Labs is more focused in research areas like object-oriented analysis, software reusability, domain-oriented frameworks, user-programmable systems, software product line, OSS and software components (Martin 1994). Lots of activities are supported in HP Labs in this context like creating a small team who is working only on reusable libraries, building application with reuse oriented architectures and frameworks.

2.3 SOFTWARE APPLICATIONS IN HYDROLOGY DOMAIN

This section presents the status of software applications in hydrology domain in different areas.

Rivwidth is a tool to automate the raster based classification for the calculation of river width from remotely sensed image. It computes the total river width along each orthogonal as shown in figure 8. Channel mask and river mask are two initial input required in the form of file to Rivwidth tool and it provides the five output files, image file along with a width spread sheet that show the total flow width of the river and other intermediates files are distance map, initial centerline and final centerline. Initial versions are very slow, after rewriting the code of computation of distance map, tool performance becomes much better. Earlier 45 minutes were required to get output on Ohio River test dataset on a Macbook pro 2.4 GHz with 2 GB RAM. but after rewriting the code approximate 17 minutes are required to get same output (Pavelsky and Smith. 2008).

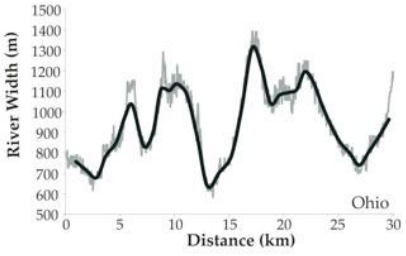


Figure 8 : River Width Calculation Output of RIVWIDTH Tool

In **early-warning and hydrological forecast system**, GIS technology was extensively used in hydrology domain. Many GIS software are developed by industry and academic each has their own mechanism to store data. However, the main obstacle is to share continentally heterogeneous nature of GIS data in hydrology domain. This obstacle is overcome by using technological solutions like data warehouse and Service–Oriented Architecture (SOA) which is designed and used along with MapGIS. This is a new method to build digital hydrology pplication with the GIS technology (WenYou and Yan. 2009).

After traditional client server, internet and World Wide Web, grid computing emerged as another network platform for computer applications. Of late, computing paradigms are start shifting towards cloud computing . Now days' use of cloud computing in enterprise applications is very common. Cloud computing mainly deals with integration of all the resource on network like computing capacity, software, information, storage, communication system, database, knowledge and main memory. Cloud computing is huge resource sharing platform over the internet. There are a few applications of grid computing in hydrology domain for fluid mechanics and interrelated fields. Distributed flood forecasting system is designed to share network resources through grid computing. Users access this system through web browser. Results of distributed flood forecasting system can be shared by state and center govt. organizations for better disaster management (Ho-Cheng et al. 2004). There are yet not any applications on cloud for hydrology domain.

A **framework** is designed for hydrology modeling system to access watershed graphical database over the network and retrieve hydrological parameters to calculate runoff hydrograph. IBM personal computer system with windows client as operating system and server with window NT are used for this purpose. Remote sensing and GIS raster technology extensively used in this hydrology modeling system (Cheng-Hua and Leu, 1998).

In **digital watershed,** integration of hydrological modeling, planning and management have many complex issues. Information sharing in digital watershed is very important for modeling assessment, management, collaboration and planning to achieve this goal an intelligent agent based server and SOA based proposed and designed for digital watershed (Shanzhen and Xiang, 2010).

The **DUET-H/WQ** "Data Uncertainty Estimation Tool for Hydrology and Water Quality" is used for estimation for uncertainty measures in hydrologic and water quality data. This will improve monitoring decision making, design, modeling and authoritarian formulation. DUET-H/WQ software calculate uncertainty estimates based upon the root mean square error propagation methodology. It improves the modeling process, environmental monitoring, and DSS-P. Hydrology and water quality data analysis used by scientist, researcher, hydrologist and water manager. DUET-H/WQ was developed by research team of Texas A&M University System through visual basic and easily distributed on windows platform and require Microsoft .net framework with service pack 2 (Harmel et al. 2009).

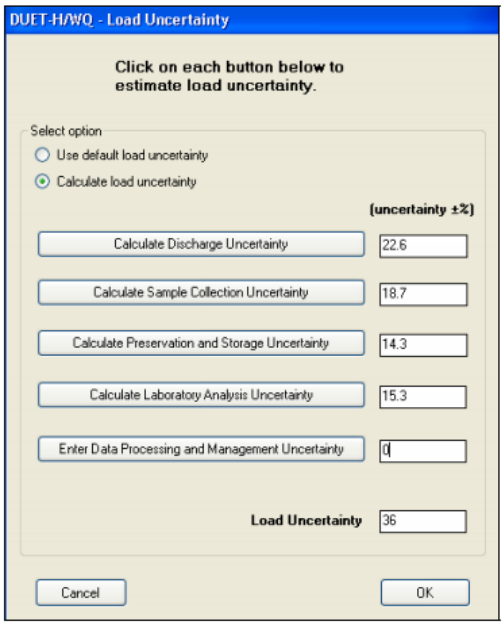


Figure 9: Calculation for load uncertainty through DUET-H/WQ

Electronic Data Exchange Model designed and implemented in China to store and exchange hydrological data so that hydrological staff have easy access of real time and accurate hydrology data. EDI model is designed on the technology base of XML and TUXEDO (Transaction for UNIX has been Extended for Distributed Applications). EDI Model is adapted and it can be very useful for developing countries to access data form heterogeneous sources (Huang and Quyang, 2010).

WaterOneFlow Web Service provides by CUAHSI to return uniform data regardless type of data repository so that end user gets uniform data. Output format is designed as CUAHSI WaterML. Main functions of WaterOneFlow are GetValues, GetVariable and GetSite etc. WaterOneFlow web service provides uniform view of heterogeneous data sources in hydrology domain (Beran and Goodall, 2009).

HydroTS is based upon the principle of SOA and SCADA to solve the complexity due to heterogeneous nature of hydrological telemetry system based upon network. Software model of hydroTS is based upon multiple layers like presentation, data resource, business and service (Ping et al, 2003).

This chapter provided the review of literature on related and relevant areas of component based software engineering, software reuse and software applications in Hydrology domain. Next Chapter presents a detailed analysis of various existing software component models.

CHAPTER 3

ANALYSIS OF EXISTING CBSE MODELS

Component-based Software Engineering is a growing area in software engineering. Component-based Software Engineering approach of software development is based upon reuse of the existing software components. It includes component requirement, their selection, integration and deployment.

There is tremendous expansion of software industry and nowadays software are too complex, too large, domain specific, the user requirements keeps on changing and there is a demand of quick delivery. In this type of scenario, CBSE can play an important role in software development process. CBSE provides the techniques for shortened development cycle, reduced development cost, improved software quality and possibly be able to solve the problem of software crisis (Bachmann et al, 2000; Heineman and Council, 2001; Szyperski, 2002).

Software reuse through CBSE deals with two aspects, one is *development for reuse* and second is *development with reuse* (Vliet: 2008).

3.1 SOFTWARE COMPONENTS MODELS

There are many definitions of software components exist in Component-based Software Engineering, still it is struggling for universally accepted one (Broy et al,1998). Widely accepted definition of software components are:

"A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties" (Szyperski, 2002).

"A component is a software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard" (Heineman and Councill, 2001).

The most acknowledged vision of software component is that it is a software unit having

- Programs to perform set of services
- Well defined interfaces to access these services i.e. requires interface and provides interface

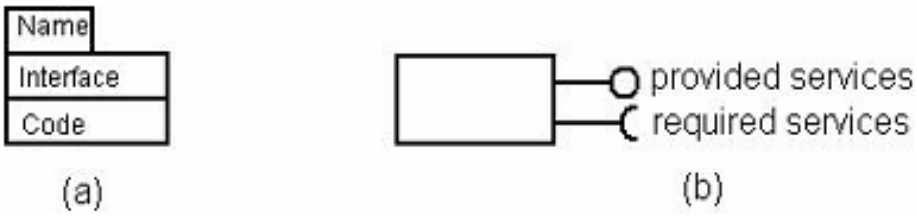


Figure 10: Software Components

Software components can be accessed through their respective interfaces as depicts in figure 11. There are two types of interfaces.

Required Interface: Required interface specifies what type of services is provided by other components. Without this interface a software component will not work. Required interface is indicated through a semicircle.

Provided Interface: This indicates type of service provided by a software component. Its specifications are based on component API. Provided interface is represented by a circle at end of the line emanating from the software component.

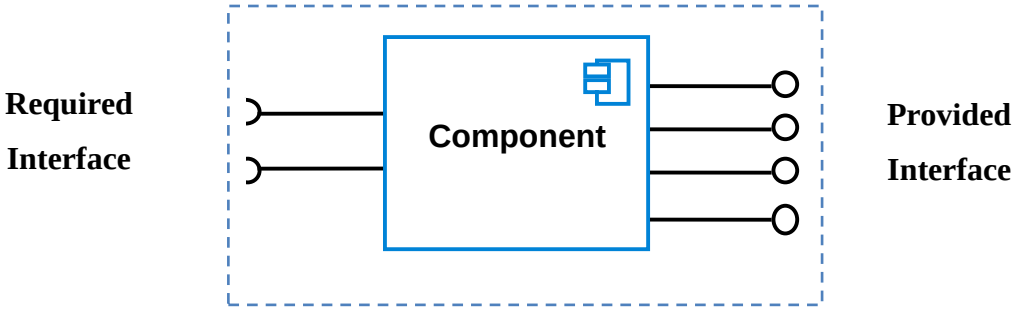


Figure 11: Component Diagram

Lau (2006) gave following main characteristics of software components:

- Information hiding is the main principle of software component construction, that's why sometimes it is known as data capsule.
  - Software component can be implement in any of the languages, not restricted to object oriented languages. Even software components can be developed using procedural languages.
  - Software components are black box in nature and provide the services from the interfaces. Interfaces may be written in IDL or have text interface interactive tools to handle services & specifications.
  - Now-a-days framework of software components support the enabling technology of plug & play software to exchange the information (Kung-Kiu 2006).

Software component model determines what a software component is and what not a software component is. Heineman and Councill provided the definition a component model as follows:

"A component model defines a set of standards for component implementation, naming, interoperability, customization, composition, evolution, and deployment." (Heineman and Council et al. 2001).

As per the definition, software component model covers all aspects like development of independent software components as well as development of information systems through software components. Sommerville(2010) specify basic elements of software component model as shown in figure 12.

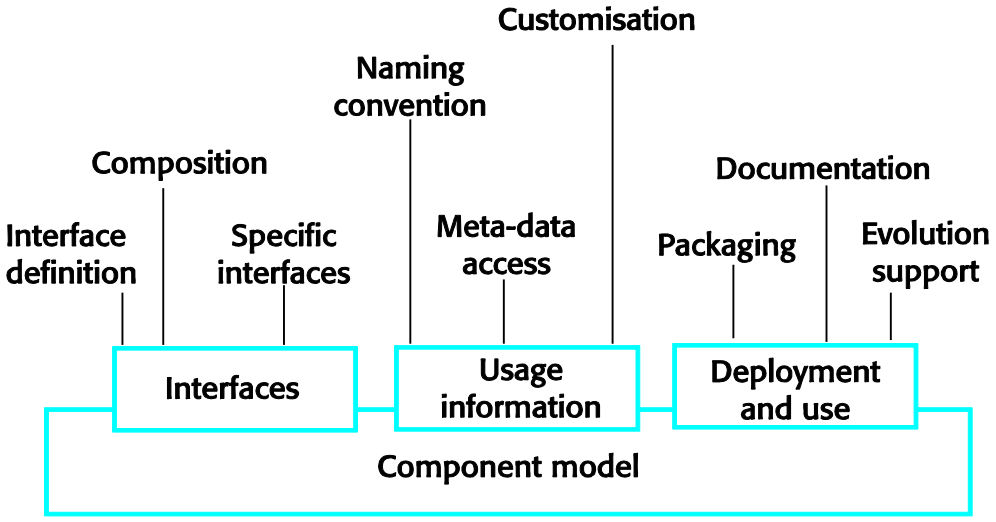


Figure 12: Basic elements of Software Components Model

3.2 CBSE BASED SOFTWARE DEVELOPMENT PROCESS

In Component-based Software Engineering, software components are building blocks for software application, greatly enhance the degree of software reusability. Software components may be OSS, COTS or in-house developed. CBSE is more focused upon re-use rather then re-inventing the software solutions (Szyperski, 2002). According to Sommerville, essentials of CBSE development process are:

- Software component should be independent such that if software component is removed of changed, then system needs not be affected.

- During integration of software components, proper standardization and specification should be opted in software components where different programming languages are used.
- Middleware supports the software components integration in distributed environment. (Sommerville,2010).

Figure 13 describes the CBSE based software application development process.

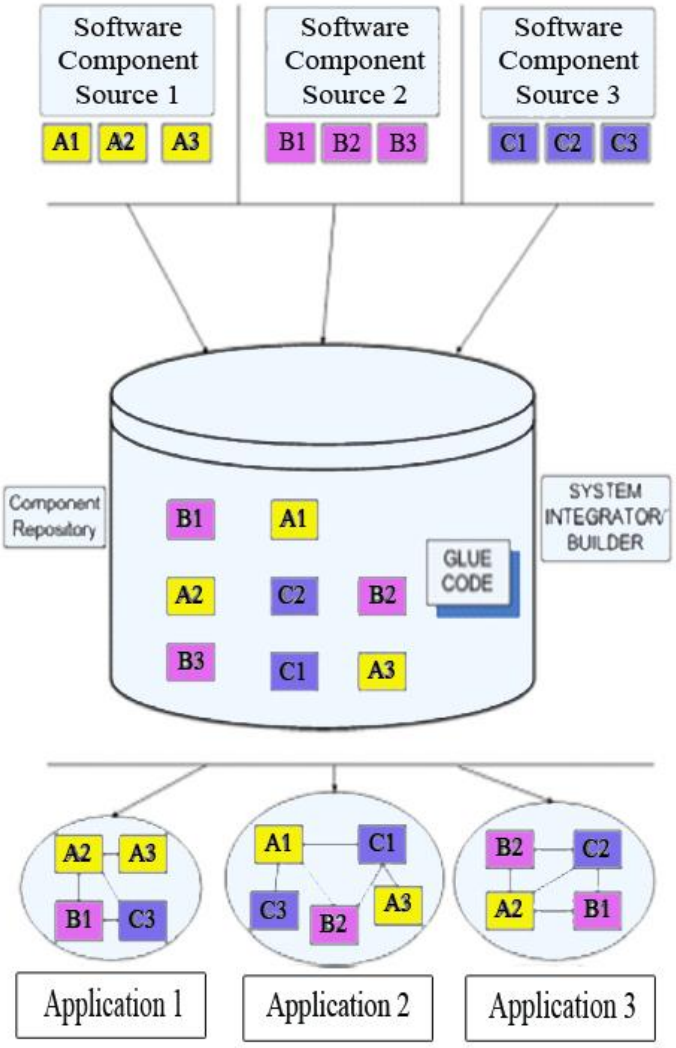


Figure 13: CBSE Based Development Process

3.3 TRADITIONAL VS COMPONENT-BASED SOFTWARE DEVELOPMENT APPROCH

Component-based software engineering approach is different than traditional software development approach. Table 1 shows the difference in these two approaches.

Table 1: Analysis of Traditional and CBSE Approach

Sr. No	Traditional Software Development Approach	Component-based Software Engineering Approach
1.	Partially focus on Software Reusability	Focus on Software Reusability
2.	Building software application from scratch.	Developed software application from existing software components
3.	Focus only on development of software application.	Development of software component as well as software application from reusable software components.
4.	In Software development life cycle, there is no special stage like selection and evaluation.	Software component selection and evaluation are special stages in CBSE life cycle.
5.	Most of the efforts on software application development.	Most of the efforts focus on selection, testing, integration and verification of software components.

3.4 RISKS AND CHALLENGES IN USING CBSE

Motivation behind the Component-based Software Engineering is reduced development cost, time and quickly fulfills the need of users. It seems to be very promising, still it a new discipline and there are number of risks and challenges are associated with CBSE. Main risk are like if software component supplier stop

producing or stop providing the software support then what happens, how can the quality of software components be assured. There are many benefits of CBSE, but the risks and challenges in introducing CBSE are to be addressed specifically. Potential risk and challenges are given in table 2 (Crnkovic 2003; Crnkovic and Larsson 2002b; Heineman and Council et al. 2001).

Table 2: Summary of CBSE Risk and Challenges

Risk/ Challenges	Description
Requirement Satisfaction	Software component selection is an iterative process and result depends upon selection process. Many software components exists in component repository, selection of right type of software components that fulfill the user need is a big challenge.
Unit Testing and Integration testing	Each software components should go for individual testing and verification as well when it is integrated with application, testing should also done with integration with other software components.
Software Tools	The area of CBSE is still not so rich in term of providing software tools for deployment, repository, selection and testing and workflow management of software component usability process.
Software component maintenance cost.	Software components are used in many applications, maintenance cost of software components is very high in each application.
Ambiguous and unclear user requirement	Software reusable components are initially designed for one application, later it is used in many software applications. All applications are unique in their nature so sometimes it is impossible to satisfy the need of all applications.
Trustworthiness of software components	Software components may or may come with source codes, mostly they are in black box in nature. So sometimes it is difficult to trust on software components.
Non isolation of Component error	Software component error are effect the performance of entire software application. Error in software components are not isolation in nature.

3.5 APPLICATION DEVELOPMENT LIFE CYCLE USING CBSE

Most significant of the challenges related to CBSE life cycle are testing and maintenance, as these are costly affair, then software component error isolation is difficult, and tuning of software components is also not easy (Gao 2000; Voas 1999; Voas and Miller, 1995). Traditional software development life cycle are not sufficient for CBSE and few researchers have focused on CBSE life cycle (Crnkovic et al, 2005). Three Component-based software development life cycle approaches are discussed next.

3.5.1 SEQUENTIAL LIFE CYCLE

Sommerville (2010) provided a sequential approach for developing applications through CBSE. There are total six steps in this approach as shown in figure 14.

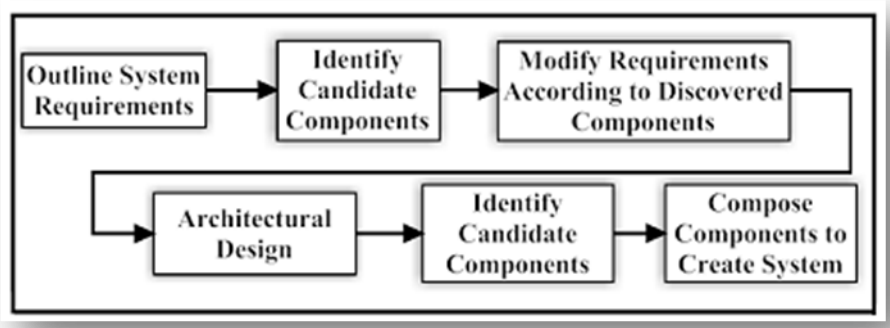


Figure 14: Sequential Approach Software development through CBSE

3.5.2 CHESSMAN COMPONENT LIFE CYCLE

Cheesman also proposed another life cycle model to develop software components with following steps, requirement, design, implementation, deployment and execution. Figure 15 shows component lifecycle. Lower half of the diagram list the way how software component are represented in particular stage of a component development life cycle (Cheesman and Daniels. 2000).

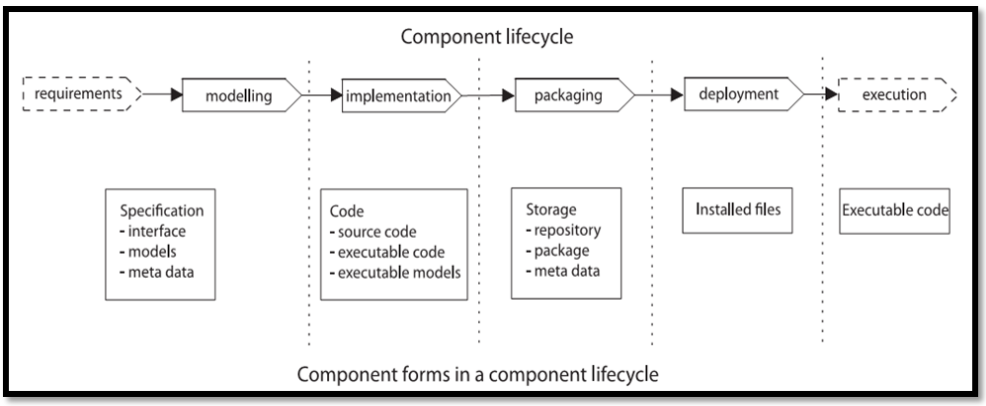
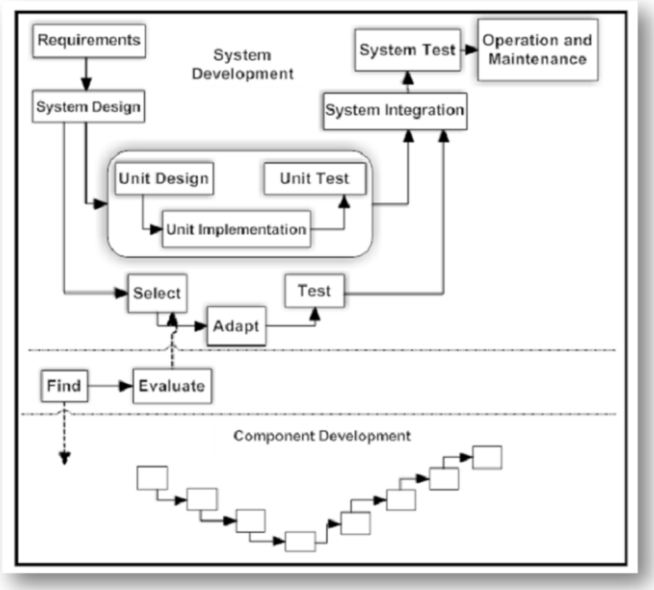


Figure 15: Component Development Life Cycle.

3.5.3 "V" DEVELOPMENT LIFE CYCLE

Ivica Crnkovic specified in his work in context of software reusability that building software systems form pre existing software components is one of the main theme of CBSE approach. He proposed "V" development life cycle for component based development as shown in figure 16.



.

Figure 16: "V" CBSE based Software development life cycle

.

Major steps in "V" lifecycle model are: Requirement, System Design, Unit Implementation, System Integration, System Test, Operation and maintenance (Crnkovic 2003; Crnkovic et al, 2005)

3.3.4 ANALYSIS

Table 3 presents a comparison of above mentioned three component based development life cycles with respect to challenges and risk described in section 3.3 and shows how each life cycle handles these risks and challenges.

Table 3: Risk and Challenges addressed by CBSE Lifecycle Process

Risk and Challenges	Component-based Software Engineering based application development Life Cycle		
	Sequential process by Sommerville	"V" Development Process	Cheesman's Development Process
Requirement Satisfaction	Partially	Partially	Partially
Unit Testing and Integration testing	Reasonable	Yes	Yes
Software Tools	No	No	No
Software component maintenance cost.	No	Partially	No
Ambiguous and unclear user requirement	No	Yes	Yes
Trustworthiness of software components	Reasonable	Yes	Yes
Non isolation of Component error	Reasonable	Yes	Yes

3.6 SOFTWARE COMPONENT MODEL CLASSIFICATIONS

Many attempts have been made on classification and analysis of software components models. Earlier classifications have been done on software components and their interfaces (Yacoub et al,1999), architecture description languages -ADL (Medvidovic and Taylor 2000), software component model (Lau and Wang 2007), added functional properties (Crnkovic et al 2005) and study of software component models in particular business area (Kotonya et al. 2003). Another work presented by Yacoub (1999) does not used any particular software component model, but elaborates on consumption and reutilization of software components. Interface classifications are based upon application interfaces that is how two software components can communicate with each other and classifications based on platform interfaces deals with interaction between software component and a particular platform.

Another classification includes software component model identification based upon set of properties like reuse, interoperability, visibility, granularity, state and use of Object Oriented approach. This type of classification is done only by business solution software components (Fellner and Turowski 2000).

Software component models are also classified based on basic properties of ADL like constraints, semantics, non functional properties and evolution (Medvidovic and Taylor 2000). Knowledge based classification of model was proposed by Kotonya where research categorized different models on several aspects like process, product, role, business domains, COTS and technology (Kotonya et al. 2003).

ACME notations are used to describes interface, assembly of software components and implementation for software component models like COM, .NET, CCM and OSGi (Crnkovic and Larsson 2002a).

Lau and Wang (2007) surveyed and analyzed 13 software component models and provided a taxonomy based desiderata for CBSE based development. He categorized software components based on composition like:

- Design without Repository
- Design with Deposit-Only Repository
- Deployment with Repository
- Design with Repository

Figure 17 depict the categorization of software components by Kue Lau.

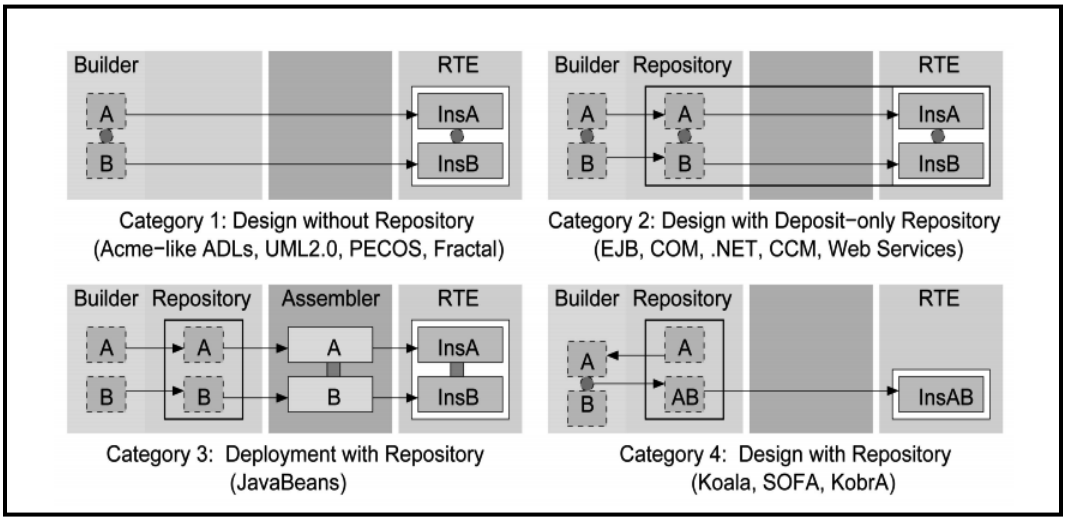


Figure 17: Kue Lau Categories based on composition

For our review and analysis, I have selected 32 software component models. Few are used in software development, whereas few are used in R& D activities and some are only used for demo purposes to express a research idea.

- Zope Component Architecture (ZCA) (Baiju, 2009).
- SOFA (SOFtware Appliances) (Plasil et al., 1998).
- SAVE Components Component Model (Kerholm et al., 2007).

- Rubus Component Model (Hanninen et al., 2008).
- Robust Open Component Based Software Architecture (Maaskant, 2005).
- PROGRESS Component Model (Sentilles et al., 2008).
- PErvasive COmponent Systems (Genssler et al., 2002).
- Palladio Component Model (Becker et al., 2007).
- OpenCCM (Contreras et al., 2012).
- OpenCOM (Clarke et al., 2001).
- Open Services Gateway Initiative (Zhou et al., 2010).
- Open MI (Moore 2010a).
- Microsoft Component Object Model (Wigley et al., 2003).
- Komponentenbasierte Anwendungsentwicklung(Kobra)(Atkinson et al,2001).
- Java Beans (Oracle 2012; Englander, 1997).
- Fractal (Bruneton and Coupaye, 2003 ;Bruneton et al., 2004).
- Fawkes Component Model (Niemueller et al., 2010).
- Enterprise JavaBeans (EJB) (Huang et al.,2009 ; Emmerich,and Kaveh 2002).
- Earth System Modeling Framework (ESMF) (Hill et al., 2004) (Collins et al., 2005).
- CORBA Component Model (CCM) (Natan, 1995;Schmidt, 2000).
- CompoNETS (Bastide and Barboni, 2004).
- COMponent-based design of software for Distributed Embedded system version II (Ke et al., 2007).
- Common Component Architecture (CCA) (Armstrong, 2005).
- C[K]omponent Organizer and Linking Assistant (Koala) (Ommering et al., 2000).
- BlueArX (Kim and Rogalla, 2009).
- Behavior, Interaction, Priority (BIP) (Basu et al., 2006).

- AUTomotive Open System ARchitecture (AUTOSAR) (Fürst, 2010).
- Web Services (Alonso et al., 2004).
- UML2.0 (Cheesman and Daniels, 2000), (Emadi and Shams, 2008).
- .NET Component Model (Kum and Kim, 2006 ; Lin et al., 2006).
- Orca(Brooks et al, 2005).
- PIN (Hissam et al., 2005).

Our classification is neither focused on successful or non successful software component models nor on business perspective. The purpose of our review and analysis is to compare these component models on technical grounds. Software component models, I have chose for our research work fulfilled the need. However, few software component models were not considered because of lack of sufficient details and technical documentation.

Software component models in different domains have been classified and analyzed on the following parameters:

1. **Interface Specification and Language Support**

2. **Nature of software component model - Domain Specific or Normal**

3.6.1 INTERFACE SPECIFICATION AND LANGUAGE SUPPORT

Software components are black box in nature, software developer use these software components models through interfaces and interfaces are mandatory part of software component specification.

Main interface languages are like UML, C, Java, IDL, Microsoft IDL, XML Pascal etc. are used details are given in table 5. Most of software component models are supported by C Language.

Table 4: Interfaces Languages

Sr. No	Interface Language	Software Component Models
	C Language	AUTomotive Open System ARchitecture (Autosar), COMponent-based design of software for Distributed Embedded Systems, version II (COMDES II),Pin Component Model.
2	CORBA IDL	CORBA Component Model (CCM), CompoNETS
3	Java Language	Java Beans, Open Services Gateway Initiative (OSGi), SAVE Components Component Model, Entreprise JavaBeans (EJB)
4	Microsoft IDL	Microsoft Component Object Model, OpenCOM
5	UML	**K**omp**o**nenten**b**asie**r**te **A**nwendungsentwicklung (Kobra), Palladio Component Model
6	XML	BlueArX, PROGRESS Component Model
7	CCA Scientific IDL	Common Component Architecture (CCA)
8	CCL-Component Composition Language	Pin Component Technology
9	Coco, Prolog	PErvasive COmponent Systems (Pecos)
10	IDL,CDL	C[K]omponent **O**rganizer **a**nd **L**inking **A**ssistant (Koala)
11	IDL, Fractal, ADL, Java or, C	Fractal
12	PYTHAN BASE	Zope Component Architecture (ZCA)
13	Robocop IDL	Rubus Component Model
14	C,Pascal	Earth System Modeling Framework (ESMF)

On the basis of interface specifications, software components models have two types of interfaces, operation-based and port-based. Operation-based interfaces have methods with parameters as a interface signature and port-based interfaces have input and output port for communications. Most of software component

models for embedded system having port based interface. Few software components model provides both type of interfaces.

Table 5 : Interface Type

Interface Specification	Software Component Models
Port Based	BlueArX, COMDES II, PIN, PECOS, SaveCCM, PRBUS, ROBOCOP, ProCOM,
Operation based	EJB, Fractal, Koala, KobrA, Java Bean, MSCOM, OpenCOM, OSGi, Pallado, Sofa
Both	CompNETS, CCM, BIP,AUTOSAR.

3.6.2 DOMAIN BASED CLASSIFICATION

Our component analysis framework is initially based upon domain specification. System requirements vary from domain to domain. One Software component model may not be fit for other domain applications. In this analysis, first component models specific to a domain are reviewed. Normal or general-purpose software component models are used to develop all type of software application independent of any domain. 32 software component models were analyzed and details of these are provided in the table 6.

Table 6:. Classification of Component Model Domain Specific / General

Sr. No.	Software Component Model	Type
1	Zope Component Architecture (ZCA)	General Software Component Model
2	SOFA (SOFtware Appliances)	General Software Component Model
3	SAVE Components Component Model	General Software Component Model
4	Rubus Component Model	Domain Specific Software Component Model

Sr. No.	Software Component Model	Type
5	Robust Open Component Based Software Architecture	Domain Specific Software Component Model
6	PROGRESS Component Model	Domain Specific Software Component Model
7	PErvasive COmponent Systems (Pecos)	Domain Specific Software Component Model
8	Palladio Component Model	General Software Component Model
9	OpenCCM	General Software Component Model
10	OpenCOM	General Software Component Model
11	Open Services Gateway Initiative (OSGi)	General Software Component Model
12	Open MI	Domain Specific Software Component Model
13	Microsoft Component Object Model	General Software Component Model
14	Komponentenbasierte Anwendungsentwicklung (Kobra)	General Software Component Model
15	Microsoft Component Object Model	General Software Component Model
16	Fractal	General Software Component Model
17	Fawkes Component Model	Domain Specific Software Component Model
18	Enterprise JavaBeans (EJB)	General Software Component Model
19	Earth System Modeling Framework (ESMF)	Domain Specific Software Component Model
20	CORBA Component Model (CCM)	General Software Component Model
21	CompoNETS	General Software Component Model
22	COMponent-based design of software for Distributed Embedded system version II (COMDES II)	Domain Specific Software Component Model
23	Common Component Architecture (CCA)	Domain Specific Software Component Model
24	C[K]omponent Organizer and Linking Assistant (Koala)	Domain Specific Software Component Model

Sr. No.	Software Component Model	Type
25	BlueArX	Domain Specific Software Component Model
26	Behavior, Interaction, Priority (BIP)	General Software Component Model
27	AUTomotive Open System ARchitecture (Autosar)	Domain Specific Software Component Model
28	Web Services	General Software Component Model
29	UML2.0	General Software Component Model
30	.NET Component Model	General Software Component Model
31	Orca	Domain Specific Software Component Model
32	PIN	Domain Specific Software Component Model

Based upon our analysis, software component models are available in following domains: Earth Modeling, Robotics, Automobiles Industries, HPC, Embedded system shown in figure 18. Remaining components are general purpose used in all type of domain applications. Chart in figure 19 show percentage of different type of component models involved in our research analysis.

\

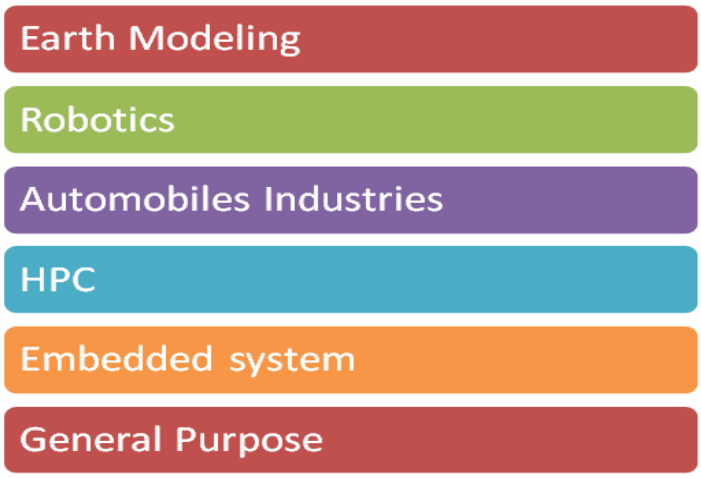


Figure 18: Components Domain

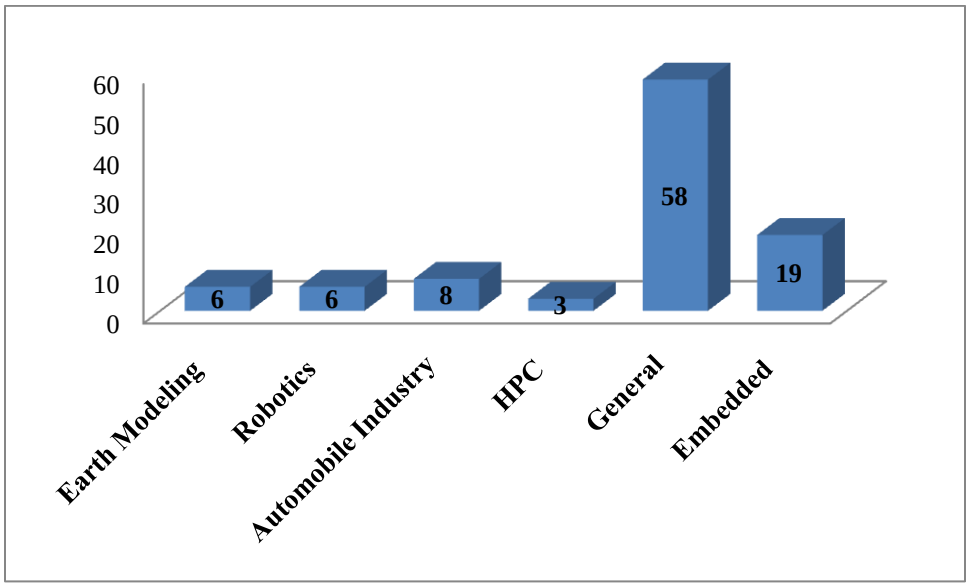


Figure 19:. Domain wise % of component models

3.7 SUMMARY

This chapter addressed some of the identified risks and challenges of using CBSE approach.

Review and comparison of software component models based on domain analysis interface specifications and language support has also been done. It was found that every software component model has its own interface specifications and few software components models are used in domains like Earth Modeling, Robotics, Automobiles Industries, HPC and most of the software component models are used in Embedded system or general purposes. The next chapter elaborates the state of art software component technologies in general and domain specific.

CHAPTER 4

THE STATE-OF-THE-ART SOFTWARE COMPONENT TECHNOLOGIES

This chapter describes the software components technologies, general purpose as well as domain specific.

4.1 GENERAL PURPOSE SOFTWARE COMPONENT TECHNOLOGIES

General purpose software component technologies may be used to develop all types of software applications to get the benefits of component-based development. Various general purpose software component technologies are

- **Common Object Request Broker Architecture (CORBA)**
- **Microsoft COM and DCOM**
- **Enterprises Java Beans (EJB)**
- **FRACTAL**
- **Web Services, etc.**

In next section we portray these general purpose software component technologies.

4.1.1 CORBA COMPONENT ARCHITECTURE

Common Object Request Broker Architecture (CORBA) is a distributed object environment that enable objects to interoperate across network, multivendor,

heterogeneous client server computing architecture. CORBA specifications are maintain by Object Management Group (OMG).Over 700 companies and organizations supporting OMG. OMG is responsible for defining and managing the standard and specification for CORBA. First version of CORBA was released in 1991 (Natan, 1995). Beside standards services CORBA offer services like Naming, Object life cycle, Events, Relationships, Transactions, Concurrency Control, Externalization and Query Supports. Interface definition language (IDL) is used by client to interact with software component on sever side written in any language like C, C++, Java, ADA and COBOL, etc. CORBA is used extensively for component based development through different programming languages along with OMG IDL.

Object Request Broker (ORB) acts as a middleware and is used to make links within software components. Figure 20 shows the remote linking through ORB client invoking the methods in the remote machine. ORB provides interoperability among different applications on heterogeneous network. In addition, CORBA component technology makes the use of XML to specify the information about software components, their packing and deployment (Schmidt, 2000).

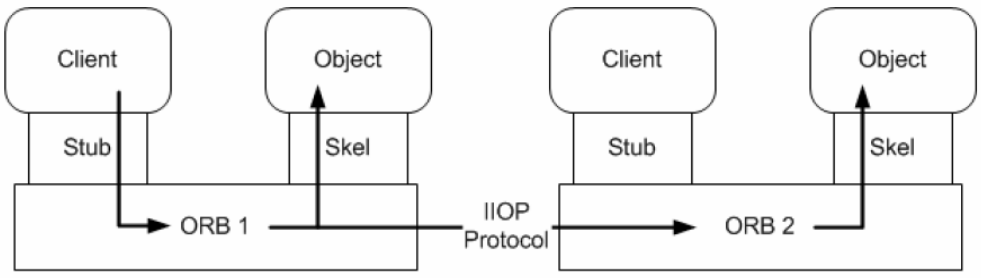


Figure 20: Role of ORB in CORBA

4.1.2 ENTERPRISE JAVABEANS

Enterprise Java Bean (EJB) is standard for developing the server side components in Java deployed in multi-tier distributed environment. Basically it comprises of one or more Java objects. Enterprise Java Beans 3.0 is one of the newly and currently used, component based platforms for distributed object-oriented applications. According to EJB specifications, bean client should deal with single exposed method in component interface. Specification requires that bean exposes a few required methods, these methods permit EJB container to manage beans uniformly without depending on the container shown in figure 21. Enterprise Java Beans run on EJB server and they are deployed in EJB containers.

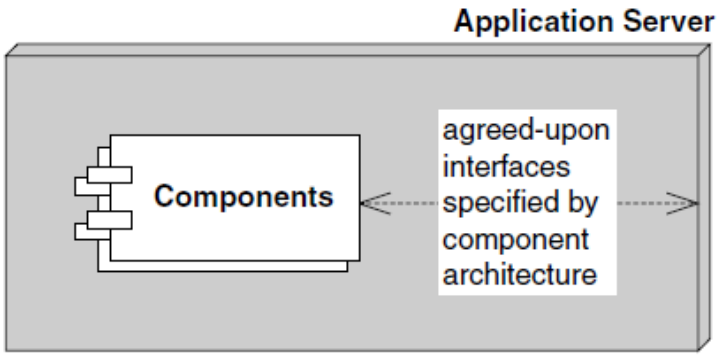


Figure 21: EJB Container

The three advantages of using EJBs are:

1- Industry Compliance
2- Portability
3- Rapid Application Development

Enterprise Java Beans are of three different types as shown in figure 22:

- Session bean

- Entity bean
- Message-driven bean.

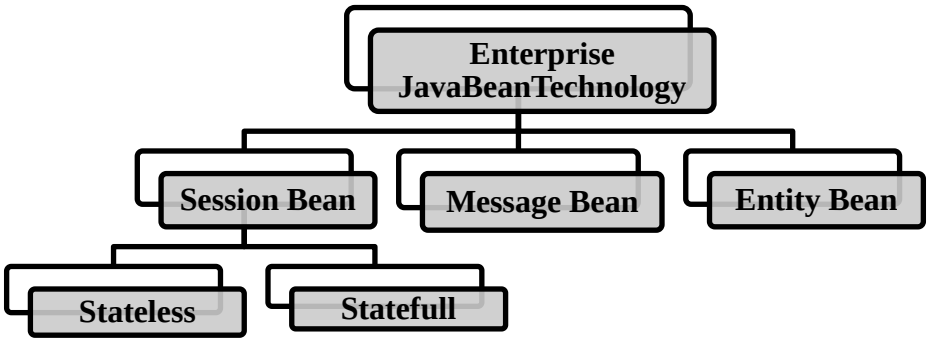


Figure 22: Type of Beans in EJB

Session beans depict business processes which are not persistent. Two different flavors of session bean are stateful or stateless beans. Stateful session beans retain client-specific session information (conversational state) operate on behalf of a single client across multiple method calls. Session remains active or alive only for a single client-server interaction session. On the other hand stateless session beans do not maintain conversational state. EJB container pools stateless session bean to handle multiple client requests.

Entity beans depict business data which is persistent and can be saved in persistent storage. Each entity bean carries its own identity with primary key. Two different flavors of Entity beans are bean-managed persistence (BMP) that manages their own persistence, container-managed persistence (CMP) that delegates their persistence to their EJB container.

Message-driven beans accept and process Java Message Service messages. They don't have any interfaces like two other beans. They do not maintain any

conversational state and they can only be accessed through messaging. Asynchronous communication is permitted between the listener and queue, which grant separation between business logic and message processing.

EJB Components play following role in a Java EE Application

- EJB components are managed by the EJB container.
- They provide business and messaging functions
- They are multi-user, secure and highly scalable

The EJB component model has the following characteristics:

- Each component is encapsulated by container
- Proxies are provided by the EJB container that allow clients limited and controlled access to the EJB component
- The proxies implement interfaces that are provided by the EJB component developer
- Clients make method calls on these interfaces

Calling EJB components from Servlets

- Servlets can freely call stateless and stateful session beans. However, the developer should observe the following guidelines in using servlets with EJB components:

Guideline	**Rationale**
Servlets should avoid calling entity classes (direct database access).	The narrowness of the transaction boundaries lead to inefficient synchronization.
Servlets should generally make a few complex method calls rather than many simple method calls.	Reduce RMI overheads.
Stateless session bean can be located at initialization time.	JNDI lookup are expensive.

Figure 23 depict the clients interaction with an EJB component system.

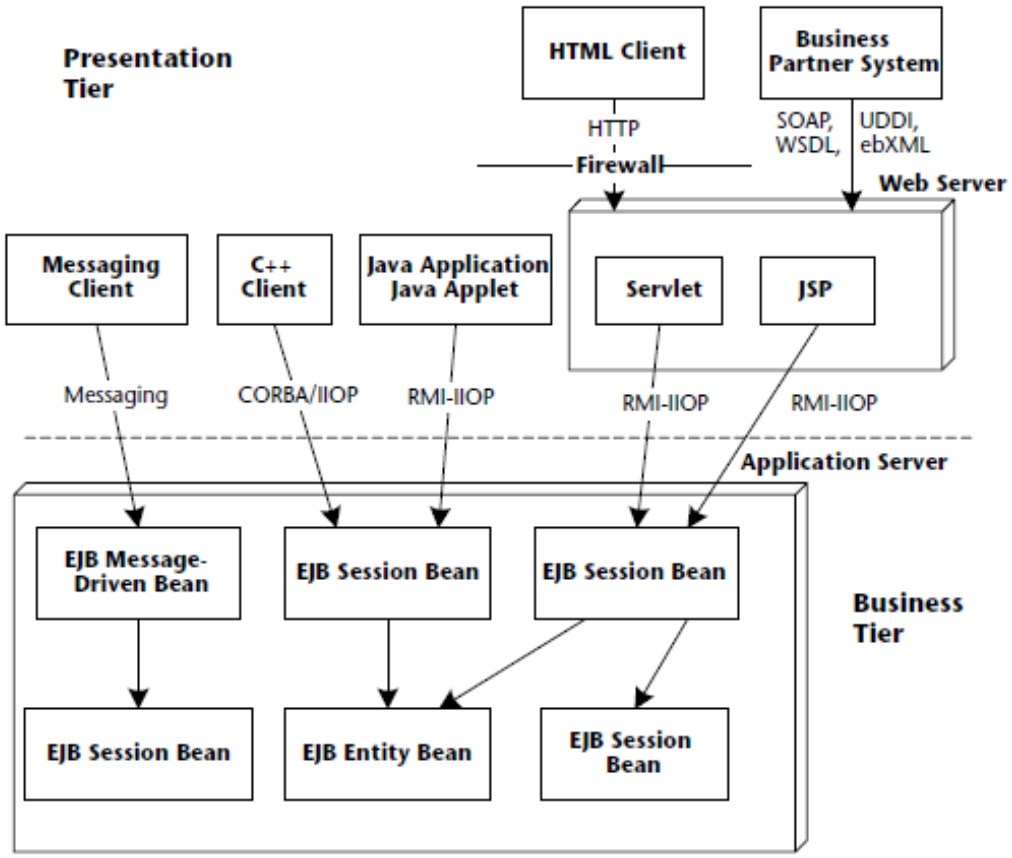


Figure 23: Clients interacting with an EJB component system

Following code shows how can a reference to a Stateless Remote Session Bean be initialized from a Servlet.

```
 private Account accr = null;

public void init() (
 try (
        InitialContext ic = new InitialContext();
        acc = (Account)ic.lookup("java:comp/env/ejb/Account");
 )catch (Exception e) (
        throw new ServletException(e);
 )
 )
```

Table 7 shows EJB component elements interface for version 3.0 of the EJB specification.

Table 7: EJB 3.0 Specification

Term	Description
Component Interface	Generic term for the business method interface regardless of it being local or remote.
Local Component Interface	Extends EJBLocalObject interface and provide a local EJB Object.
Remote Component Interface	Extends EJBObject interface and provide a remote EJB Object.

4.1.3 FRACTAL

The Fractal component model used to deploy, implement, control, monitor and dynamically configure software application. Main attributes of fractal component model are composite components, shared components, introspection and re configuration (Seinturier et al., 2006). In Fractal components, interface is access point. Two types of interfaces exist: client interface and server interface. Client interface supports the outgoing operation invocation and server interface supports the incoming operation invocation. Membrane support internal feature of components and contents represent the finite set of software components as shown in figure 24.

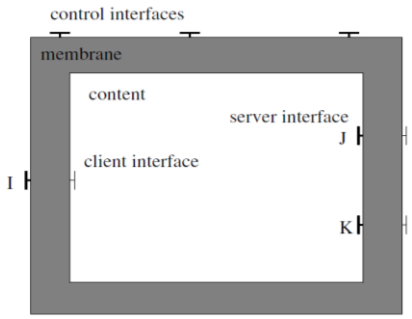


Figure 24: A Fractal Software Component

Fractal Component technology supports various type of controller like Lifecycle controller, Content controller, Binding controller and Attribute controller. Work on fractal is initiated by France telecom and INRIA Sardes in 2000 (Bruneton et al., 2006).

Implementations of Fractal components models are available in Java through AOKell, ProActive, Julia and in C through Cecilia and Think frameworks. Fractal component technology is used in transaction management, Operating system kernels, Multimedia embedded application, distributed system management and performance evaluation (Philippe, 2008).

4.1.4 MICROSOFT COM AND DCOM

Microsoft has taken the easiest route in components technologies. Instead of becoming a part of open standards initiative, it has continuously re-engineered its existing applications and platform base.

Component technology introduced by Microsoft made successes story with Visual Basic control, Object linking and embedding (OLE), Active X and COM/DCOM. COM is a Microsoft foundation on which all component software is based. COM components are made available on the Macintosh also (Box,1998).

COM is a binary standards, it specifies nothing about how a particular programming language work. In COM, collection of objects is treated as software components. Fundamental entity of COM is defined through interface. In binary view interface is represented as a pointer. COM components are free to contain any number of interfaces.

Figure 25 on next page depict the layers in Microsoft software components.

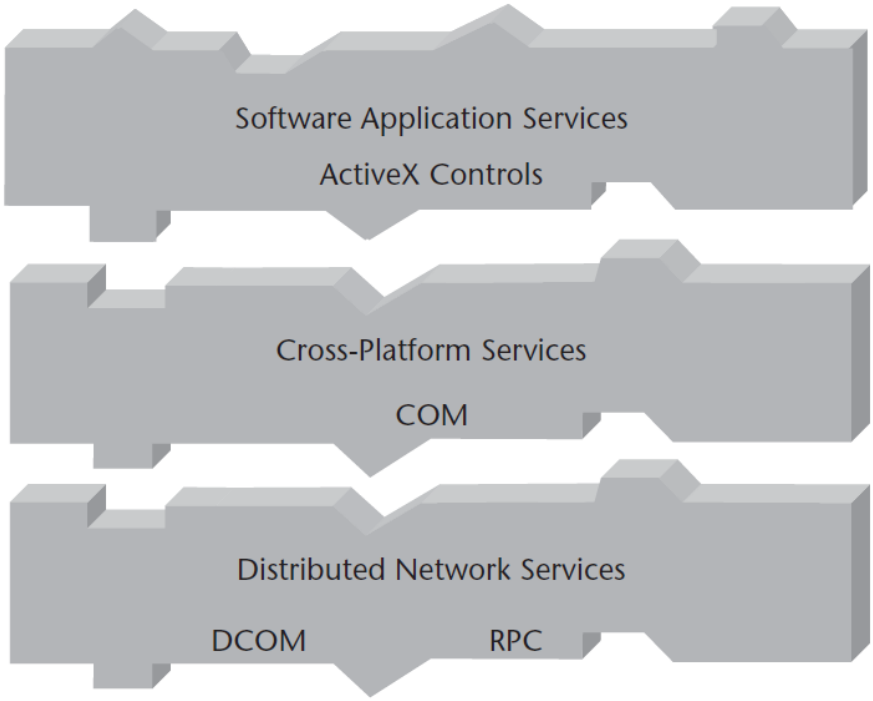


Figure 25: Microsoft component layers

Microsoft released the Distributed Component Model (DCOM) in 1995 for Windows NT so that COM can perform client server computing on the network and meet the needs of industry and be able to compatible with CORBA. DCOM architecture provides the feature of location independence and connection management (Wigley et al., 2003).

4.1.5 WEB SERVICES

In Service-Oriented computing, web services are primary elements of web computing. Web services support machine to machine communication and resource sharing. Interfaces of web services are describe through WSDL and SOAP messages for communication over the network by using HTTP along with XML specification based upon web related standards. WSDL interfaces manage data type, message format, transport protocols within many services (Newcomer,

2002). Figure 26 depict the web service and Figure 27 depict the composition of web services.

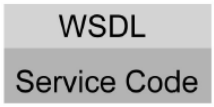


Figure 26: Web Service

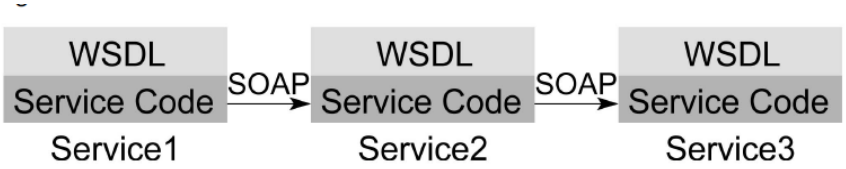


Figure 27: Composition of Web services

Web services can be design, develop, implemented in any language and deployed on the server which is available through IP address. UDDI is used to publish the interfaces of these services. Like software components, web services also have well define interfaces and services as a set of objects and respond through web. It is kind of software components based upon web technologies (Yu et al., 2007).

4.2 DOMAIN SPECIFIC COMPONENT TECHNOLOGIES

To solve domain specific problems, Domain Specific Component Technologies are used. Many software companies as well as research institutes are developing and providing support for the domain specific component technologies in the areas of Automotive, Robotics, Embedded systems and Earth Science.

This section outline the domain specific software component technologies like AUTOSAR and SaveCCM in Automotive industry, Fawkes in Robotics, KOALA and PIN in embedded system, CCA in HPC and Earth science.

4.2.1 AUTOMOTIVE INDUSTRY

4.2.1.1 AUTOSAR

AUTomotive Open System ARchitecture (AUTOSAR) was founded in 2003 and its main aim is to provide facility of reusability and exchangeability of Software components between different vehicle platforms and improve complexity management of integrated automotive electronics architectures. AUTOSAR is supported by large number of automotive, electronics, semiconductor and software companies. Software are describes in AUTOSAR as software components (sw-c) (Heinecke, 2004).

AUTOSAR supports communication types like sender-receiver, client server and AUTOSAR software component are implemented in C. AUTOSAR system architecture has four layers a). Microcontroller abstraction layer, b). ECU abstraction layer, c). Services layer, d). Application layer. In application layer Component-based software engineering is used. Overview of AUTOSAR system architecture is shown in figure 28.

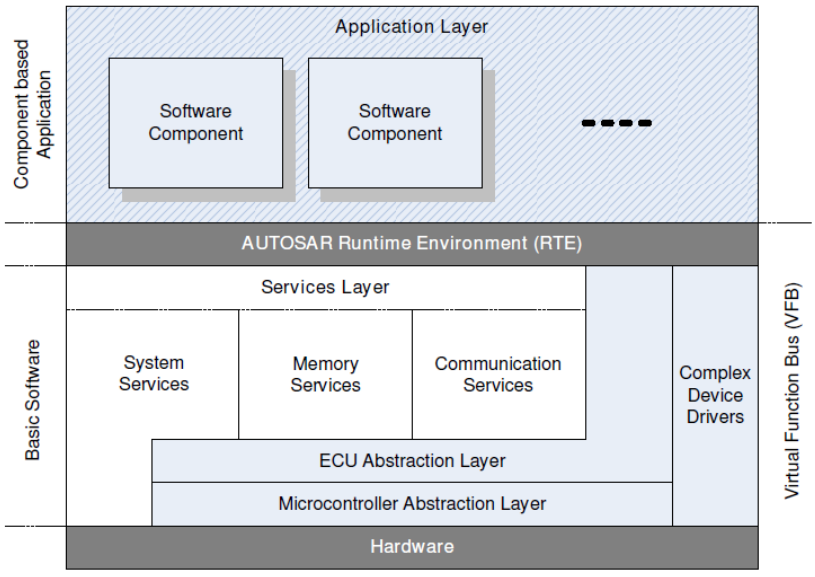


Figure 28: AUTOSAR system architecture

Now all vehicle domains are associated with AUTOSAR and AUTOSAR standard will be the platform on which future vehicle applications will be work.

4.2.1.2 SAVECCM

SaveCCM software component basically deal within the domain of vehicular systems and it is the part of SAVE project (Hansson et al., 2004). It is primarily focused on the safety critical subsystems such as the responsibility of controlling vehicle dynamics such as braking, steering etc. In today's world systems are build which include components which are based on software and computers rather than being mechanical as in earlier times **(**Crnkovic and Larsson, 2002a). The main purpose of SaveCCM is to make the design process more and more efficient while reducing the risk as much as possible. The software components in SaveCCM have following characteristics:

a) They are statistically configured software components, usage would be decided at the time of design.
b) It is of utmost importance to satisfy the attributes of timing and dependability.
c) SaveCCM software components are tested on vehicular systems and simulation techniques on computer, its resources use should be as low as possible.

SaveCCM component technologies consist of Components, Assemblies, Run-time framework, Switches etc. SaveCCM component model take care of following quality attributer during the design of software components, end to end timing, freshness of data, simultaneity, jitter tolerances (Hansson et al., nd).

4.2.2 ROBOTICS

Component Based Software Engineering (CBSE) emphasizes on the fact that a module (data + function) once created should be such that it can be customized and enhanced by the programmer whenever required in future. CBSE finds its application in development of *Component Based Robotic Software (CBRS)*. Each

component implements various interfaces. Through these interfaces the components interact with each other and thus the various units work together cohesively.

4.2.2.1 FAWKES COMPONENT MODEL

Writing the source code for any robot is a complicated and time consuming task. Certain common functionalities are implemented by all robots. Creating a different module each time to perform the same function would be prohibitively expensive. Thus the main idea behind having **component based robotic software framework** is **reusability** of the source code. The program written for one particular robot must be transferable. In order to make the source code independent of the inner details of the robot and **enable abstraction**- the code must be written against a cross platform/ framework. Fawkes is one such Component-based software framework to design real time robotics applications. In Fawkes framework, code is wrapped in a well defined API. Fawkes tries to reuse API interfaces to the maximum and emphasis a closely integrated system. Thus Fawkes enables synchronization of tasks and proves to be efficient in embedded systems as well as multi-machine service robots (Niemueller et al., 2010).

In Fawkes software components are treated as logical units. Usually one component implements a single plug-in. Data is exchanged between the plug-ins using a central blackboard as shown in figure 29.

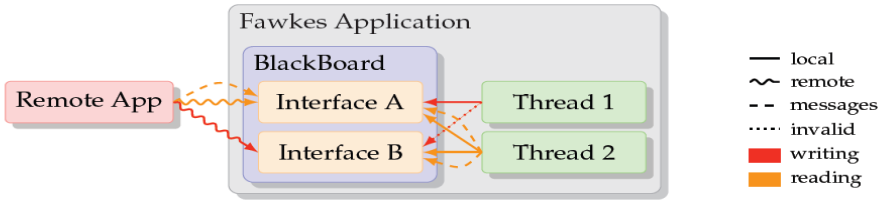


Figure 29: Fawkes blackboard

Fawkes Component based framework designed to fulfill following standard requirement software in robotics domain.

Robot hardware abstraction: The programs that run on the robots should be platform independent.

Extensibility and scalability: Additional features should be easily added to the framework in order to enhance it.

Limited run-time overhead: The memory and CPU requirements of the framework should be feasible.

Software characteristics: The software should be well defined, simple and consistent.

Tools and methods: The framework must provide facility introspection of data, tools for monitoring, or debugging.

Documentation: Well defined documentation of the API helps developers to understand the framework.

Fawkes framework is based upon agile development methodology. Fawkes framework basically targets two robotics platforms like wheeled service robots, robotic soccer domain (Niemueller et al., 2010).

4.2.3 EMBEDDED SYSTEM

Today's electronic market is incomplete without embedded software. It is used in almost all electronic products. Earlier, minor challenges occurred while developing Consumer Electronics software, but in present scenario mainly three problems in software solution for embedded system are size and complexity, increasing diversity of products and their supporting software, and the very little

time available for development. Following sections discuss the component technologies in use in embedded systems.

4.2.3.1 KOALA

Philips develop C[**K**]omponent **O**rganizer **a**nd **L**inking **A**ssistent (Koala) for development of software in consumer electronics (Ommering et al., 2000). Koala components are defined through:

1. Data Definition Language(DDL)
2. Interface Definition Language (IDL)
3. Component Definition Language(CDL)

Interface is represented through triangle. Koala Model workspace is known as the Repository for KOALA software components. In the component composition of KOALA component model three type of conector are used binding, switch and glue code. Figure 30 depicts the components composition through KOALA.

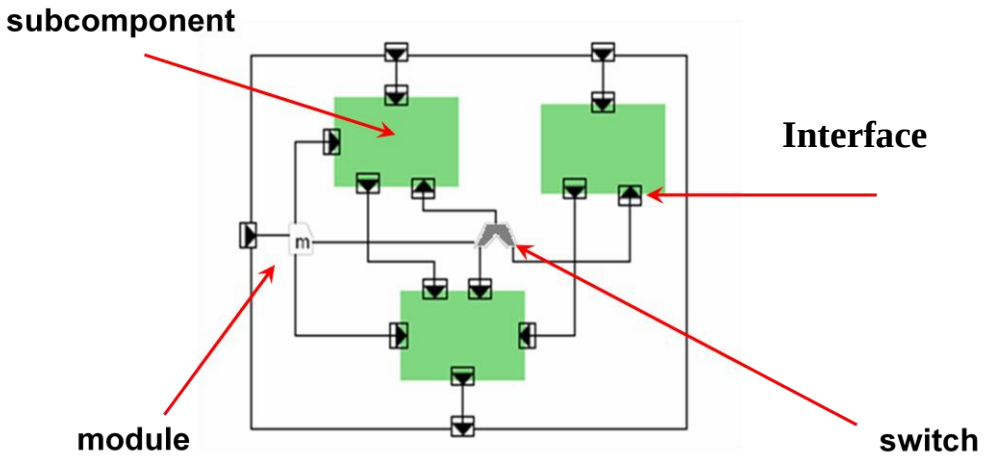


Figure 30 : Component Composition - Koala

Koala components are compiled into C language. Today Philips software developers are mostly using koala component technologies for designing software embedded system. It is used in variety of situations and very useful in developing software solution for various Consumer Electronics products (Maaskant, 2004).

4.2.3.2 PIN COMPONENT TECHNOLOGY

Pin software component technology is developed by Carnegie Mellon Software Engineering Institute (SIE) for use in prediction enabled component technologies (PECTs) designed for building embedded software applications so that it can connect components to their respective interfaces. PIN Component technology is based on two parts-component model and runtime environment. Component model describes the structure of particular component which is logical and implementable so that it can construct applications and also set some ground rules for the interaction between components and resource sharing. Further, runtime environment makes sure that the rules are followed as well as basic services like resource sharing, scheduling etc. are made available through it (Bass et al., 2005).

Construction and Component Language (CCL) is used to design components in PIN component Technology. CCL is an architecture description language in the "component and connector style" specialized to work with PIN component technology (Hissam et al., 2005). Structure of the Pin Component Technology is given in figure 31.

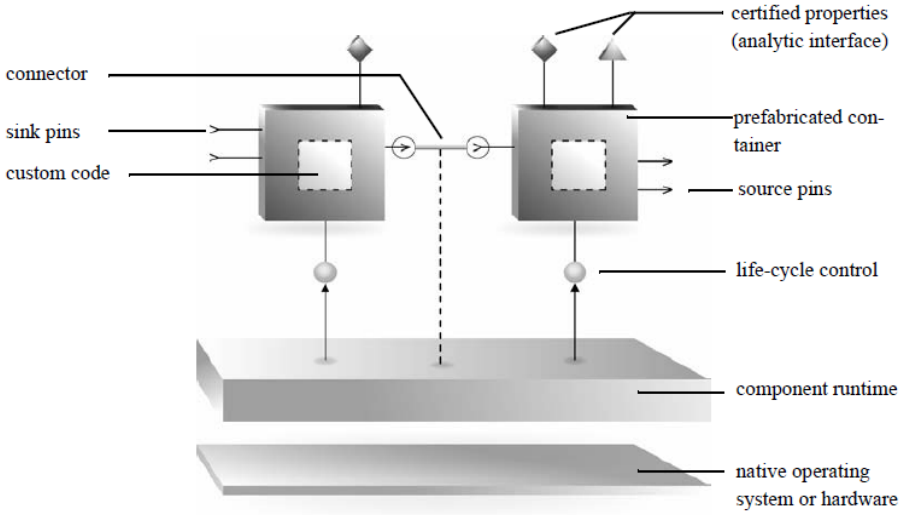


Figure 31: Structure of the Pin Component Technology

The design of PIN is based on 5 objectives which are:

1). A simple programming model and execution model supporting UML state charts.
2). Various ways to make sure that extrinsic design and implementation constraints.
3). Basic features needed to build predictable embedded software,

4) .Able to change to the needs of new applications and platforms,

5). Freely distributed.

Components of pin includes two parts i.e. a user supplied function and the pin supplied container function. Also the logical structure provided by Wallnau and Ivan is based on FIFO (first in first out). Pin V1.0 supports windows NT as operating system and uses C language interface (Ward-Dutton and Containers, 2000).

4.2.4 EARTH SCIENCE

4.2.4.1 COMMON COMPONENT ARCHITECTURE

The standard component architecture for high performance computing is Common Component Architecture (CCA) developed by National labs and group of universities for use in earth sciences (Box 1998; Armstrong and . Kumfert 2006).

Standard CCA interface is used in the framework and each component has its input and output through scientific IDL. Details of these IDL are available in CCA repository. CCA configuration APIs are used for collaboration with other components in different building environment as it is shown in figure 32 in next page.

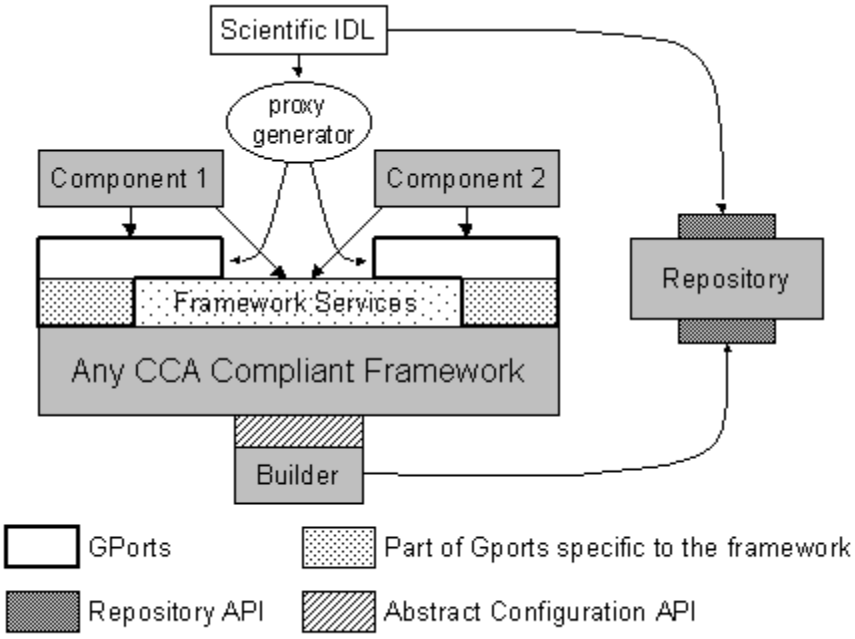


Figure 32: CCA Component Architecture

Main elements of The Common Component Architecture are CCA Scientific IDL, Gports, Framework Services, Configuration API and repository API. It functions with CORBA-like distributed-object components. CCA is used for HPC scientific computing and its application areas exist in earth science also (Armstrong and Kumfert, 2006). A number of CCA tools (CCA Tutorial Group 2010) includes following to create and run CCA Components:

1. **Ccaffeine** - CCA specification and CCA framework.
2. **Ccaffeine GUI** - Ccaffeine‘s Graphical User Interface.
3. **Chasm** - Los Alamos provide an F90 interoperability library.
4. **CCA Specification** - Specification for high performance computing components.

5. **Babel/SIDL** - an object oriented interoperability interface definition language.

Figure 33 shows component integration through ports in CCA.

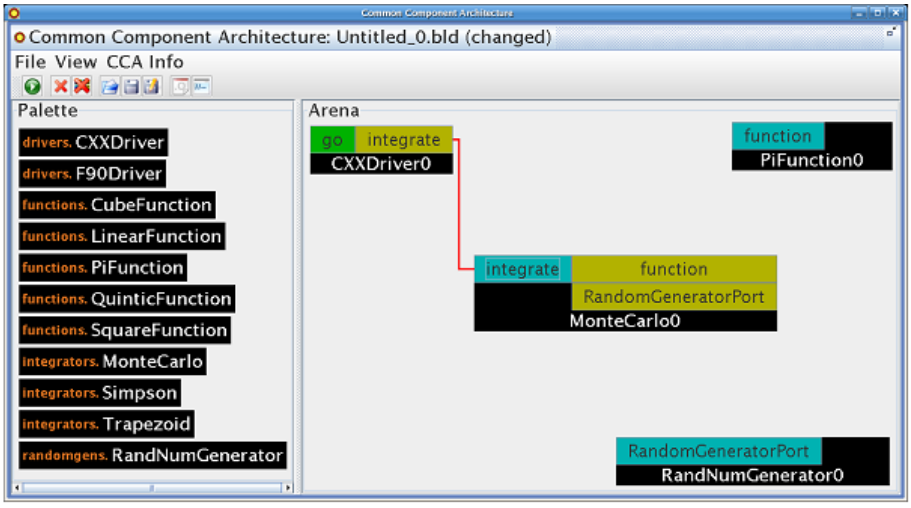


Figure 33: Integration through ports in CCA

This chapter disused general purpose as well as domain specific state-of-the-art software components technologies in use. Next chapter discusses hydrology domain and software being used in this domain.

CHAPTER 5

DOMAIN EXAMPLE - HYDROLOGY

This chapter discusses the domain example of hydrology and use of software in Hydrology domain.

5.1 THEORETICAL BASE OF HYDROLOGY DOMAIN

Science of water is Hydrology. It deals with distribution, occurrence and circulation of water found on earth and its environment. Hydrology is also concerned with water found in ice, rainfall, snowfall, lakes and below earth's surface. It is a very broad and inter-disciplinary subject and can be classified into two broad categories – scientific hydrology and engineering/applied hydrology. Scientific hydrology is concerned essentially with academic aspects of water science, whereas engineering / applied hydrology deals with engineering applications. Understanding of hydrologic cycle is essential to the study of hydrology.

5.1.1 HYDROLOGY CYCLE

Hydrology cycle is an endless cycle of water on the earth and its environment. Water in the earth is available in the atmosphere as moisture, in the sea and oceans as sea water, on the land flowing as stream and rivers, collected in ponds and lakes, within and below the soil as moisture, ground-water. Change in distribution and circulation of water above and below earth's surface and

temperature of the earth affect water resources. Solar energy causes evaporation and precipitation, making water molecules to move from earth's surface to the atmosphere and back to earth. Activities between evaporation and precipitation can be classified under transportation and storage heads as shown in table 8.

Table 8: Hydrological Components

Transportation Components	Storage Components
Precipitation **Evaporation** **Transpiration** **Runoff** **Infiltration**	Storage on the land surface, like ponds, lakes, artificial reservoirs and dams, etc. Storage below the land surface, like ground water storage and Soil moisture storage

Greek philosopher Thales c~550 BCE may have this hydrology cycle in mind when he concluded that water makes up everything in this world. Hydrology cycle traces all activities between evaporation and precipitation and can be diagrammatically depicted in figure 34.

Figure 34: Hydrology Cycle

5.1.2 HYDROLOGY SYSTEM CONCEPT

The hydrology cycle can be symbolized as a system and a sub system. Representation of hydrology as a system by Jamal is shown in figure 35. Hydrology system cycle has three subsystems: the atmospheric water sub system, the surface water sub system, and the subsurface water sub system (Jamal,2011). Key hydrological processes are precipitation, groundwater outflow, infiltration, surface runoff, evaporation, transpiration and overland flow etc.

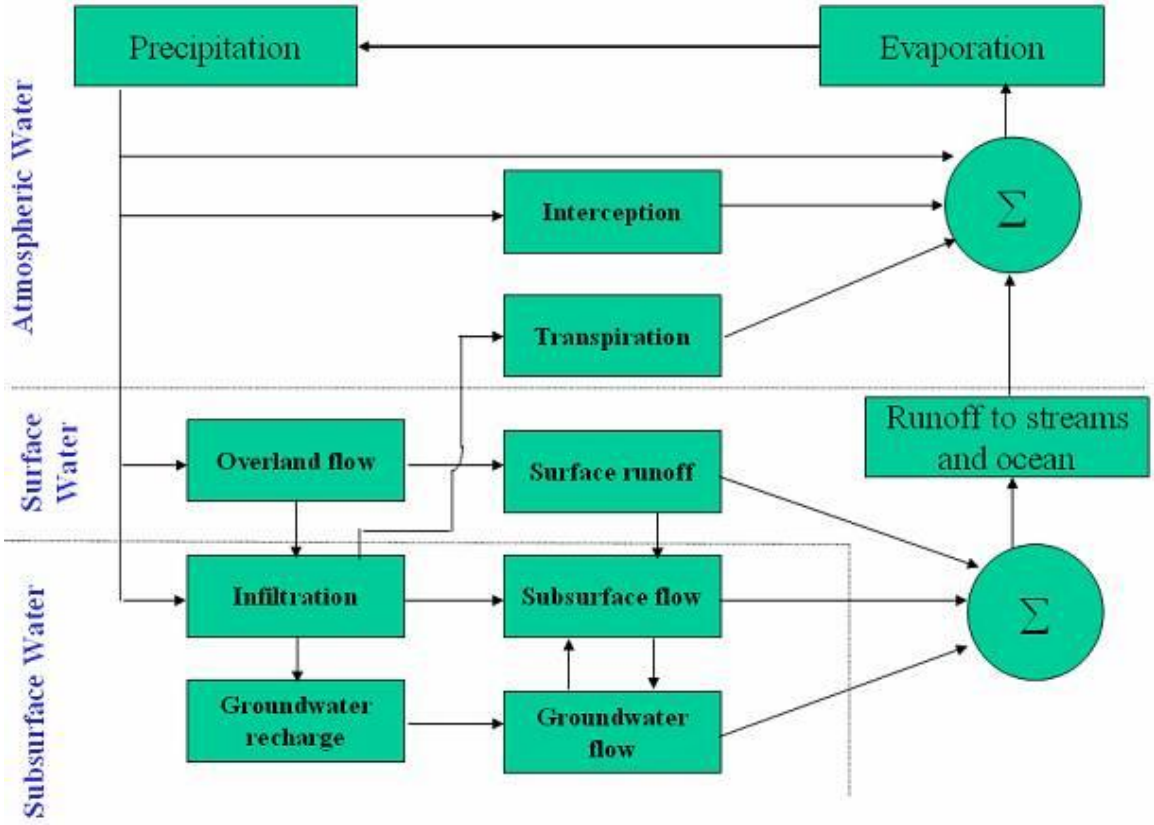


Figure 35: Hydrology System Concepts

5.1.3 APPLICATION OF HYDROLOGY IN ENGINEERING AND SCIENCE

Human civilizations from ancient times have thrived on river banks. Water is precious resource for mankind. Knowledge of Hydrology has consistently been

applied for the welfare of society, like conservation and preservation of water & water bodies to help provide water supply for drinking purposes in rural and urban areas, for large scale irrigation of fields, to control floods, to generate hydropower, to meet the demands of industry and pilgrimage centers, to protect wild life, flora, fauna and biosphere leading to conservation of ecology.

Today, modern-day science and engineering principles are being widely used by scientists/researchers, agriculturists, planners, administrators/managers, engineers, forest and army officials to achieve the aforementioned welfare activities.

5.2 HYDROINFORMATICS : INFORMATION TECHNOLOGY (IT) IN DOMAIN

Various generations of modeling in hydrology where IT, software, GIS, AI etc. have been used. Computational hydraulics introduced IT into hydrology. IT today has permeated into almost all office activities of all concerned working with state and centre government agencies dealing with water in India. The official website of Ministry of Water Resources, Government of India has more information.

5.2.1 FLOW OF INFORMATION IN HYDROLOGY

IT plays very important role in flow of information in hydrology system. Flow of information in one of IT systems being used in hydrology is shown figure 36,

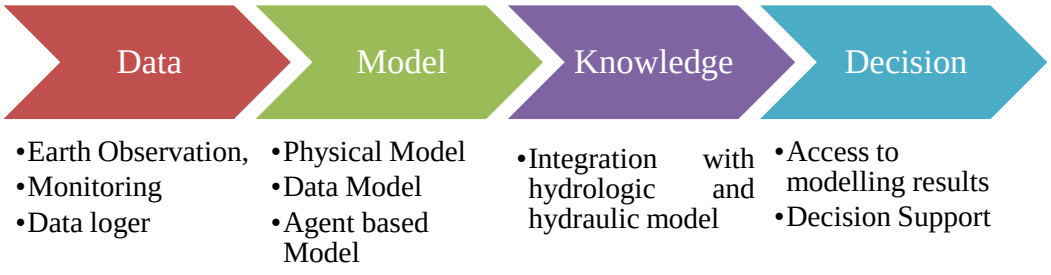


Figure 36 : Flow of Information in Hydrology System

5.2.2 DATA INTEGRATION

Hydrology related data is collected from different locations by many state and central Government organizations and is shared by many agencies. Data in scientific computing in hydrology domain is heterogeneous in nature and is collected from different sources.

Now-a-days, IT plays a very important role in data integration. For instance, Indian Meteorology department (IMD) uses the data captured by weather satellites for weather forecasting integrating it in their own forecasting models. Its output is being further utilized by other agencies in their own systems.

HIDE is a web-based data integration system that helps hydrologists and water managers to access, distribute and dynamically integrate data from different data sources rapidly. HIDE is being used for integrating hydrological data from many organizations.(Ravindran and Yao, 2007)

Another example of data integration in hydrology is provided by "The Consortium of Universities for the Advancement of Hydrologic Science, Inc" (CUAHSI), which is a group of 122 universities. Its mission is to facilitate the to provide to understanding of the central role of water to life water science community to advance, earth and society (Cuahsi, 2012). One of the main goals of their project is to provide the availability of hydrological data (Tarboton et al, 2010). CUAHSI project is funded by National Science Foundation (NSF) of Unites States of America. Hydroserver is a name given to the server system catering to data availability, data sharing and data integrate among distributed national/international hydrology databases (Horsburgh et al, 2010). HydroServer is used to integrate and publishing hydrologic data through set of commercial software applications. Figure 37 shows the architecture of HydroServer (Server, 2012).

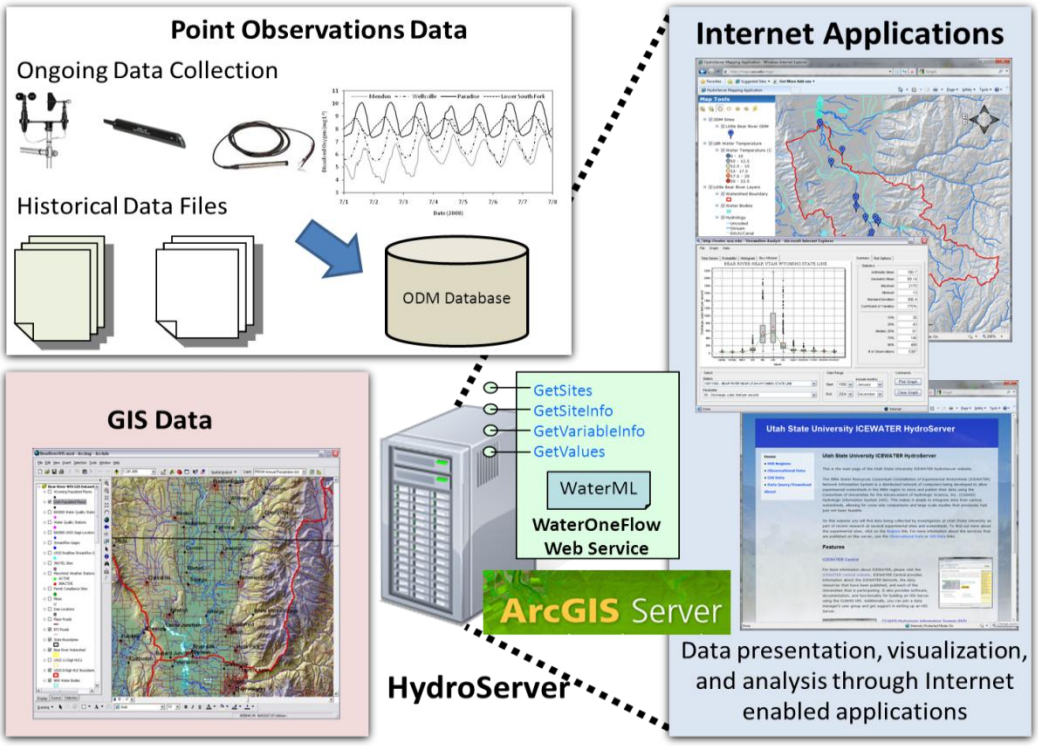


Figure 37: HydroServer Architecture

HydroServer is configured using commercial software, such as ESRI's ArcGIS Server and Microsoft SQL Server for publishing hydrologic data for research and development purpose. List of commercial software used in HydroServer are shown in figure 38.

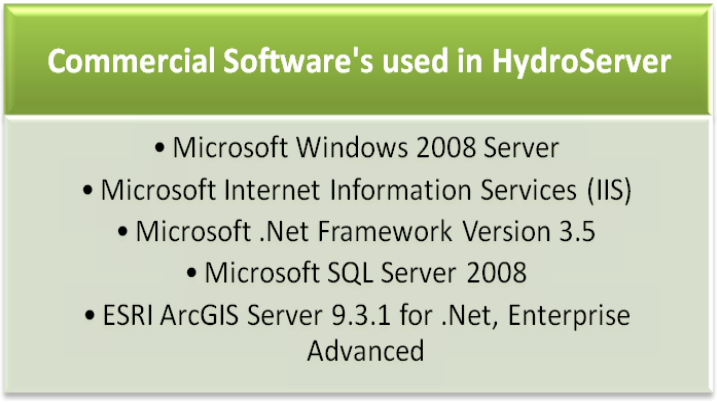


Figure 38: Commercial Software used in HydroServer

5.2.3 DECISION SUPPORT SYSTEMS

A software company DHI Software specify in its Needs Assessment Report to Government of India's for Hydrology Project II have suggested a Decision Support System (DSS) for hydrology. The report have outline many successful cases in support of the applicability of DSS as Integrated Water Resources Development and Management System. It will fulfill the requirement of all the states and centre organizations dealing with hydrology.

Any DSS applications will require easy access to hydrology data and will make extensive use of models. At the core of the DSS will be a database, containing not only time series of hydrological and meteorological data but also geo-referenced information on physical elements of the river basins.

The modeling software should be integrated with different models are used by different organizations, so that model execution and scenarios analyses can be controlled by the DSS and the results may be imported for further processing and analysis.

The main requirements for the functionality of the DSS software for hydrology must include:

Data management: the DSS must include functionality to store, organize, manipulate and display time series data and spatial data.

Model integration: the DSS must be able to handle the associated modeling software so that the user can apply the model directly to address real world problems. Import of results of other models should also be enabled.

Scenario management: definition and execution of scenarios is a key requirement of this DSS. It must be straightforward for a user to execute "what-if" scenarios when ever required.

Analysis Management: performing various types of analyses on top of executed scenarios, including economic analysis

Report Management: producing template based reports illustrating e.g. scenario output and comparisons of scenarios. The DSS should be able to adapt to the specific requirements for the actual implementation. In some cases this will involve development of special routines and methods, in other cases it may be support for special models.

DSS should be customizable, scalable and expandable with respect to at least:

Users - the system must be able to handle the number of users required, whether it is a single use system or a system serving many users.

Data - a complex setup with many models will generate a lot of data especially compared to a single user, single model system. The architecture of the system must allow configuration to handle this.

Models - it will be possible expand the system to include new model types to adapt to project requirements

Functionality - the standard functionality of the DSS must be expandable to accommodate project specific requirements. Custom displays, plotting formats, time series analysis tools, are examples of custom functionality required. Another example is the ability to disseminate data via custom web displays.

5.2.4 MODELING PARADIGM

Like in other domains, IT contributes towards computer based models, in hydrology as well. Modeling is used to illustrate the reality. Computer based modeling uses a computer program to simulate a physical model based upon particular physical process with the following two main objectives:

1). To provide clear cut understanding of particular domain through simulation, for example water distribution for irrigation, river catchments, aquifers, etc. Modeling helps to understand working of the system and its past performance.

2). To provide a platform for forecasting the results and analysis about existing water systems, for example, effects on humans by building of dams, protection of land and river from flood.

Modeling is extensively used in hydrology and will remain so in future also. There are lots of commercially sophisticated modeling packages available today developed by many organizations, like HR Wallingford [UK], Danish hydraulic Institute and Delft Hydraulics [Netherlands]. There are three main type of modeling paradigms in hydrology.

5.2.4.1 PROCESS BASED MODELING

Process based modeling is a scientific understanding of biology, physics, and chemistry of water science. Study of water science is an inter-related subject. It is also known as physical modeling, simulation or numerical modeling. Lots of emphasis is given on designing of safe and reliable software modeling systems, because it will become a basis on which real systems will be built later on. Process based modeling reduces the complexity of complex systems (Jonkers, 2001). Models are generally designed for the engineers to take appropriate decisions while building real systems. An example of process based modeling is its use in a tunnel and a bridge constructed between Denmark and Sweden

(Thorkilsen and Dynessen, 2001). Another example of process modeling is its successful use in river basin management (Falconer and Lin, 2005).

5.2.4.2 DATA-DRIVEN MODELING

Data- driven modeling is different from physical modeling. Physical modeling is based upon knowledge of physics, chemistry and biology encapsulated within a computer program. Data- driven modeling is based upon learning from available data and incorporates the unknown dependencies within system inputs and outputs. Here data means a known sample that includes both input and output (Khu et al., 2004).

Most popular data-driven modeling method is based upon artificial neural network.

In an Artificial Neural Network (ANN) based data-driven modeling, accuracy depends highly on data range and quality of data (Dibike et al., 2001). Success of ANN based data-driven modeling is based upon number of unknowns.

In data driven modeling other techniques have also become popular in recent years, like Support Vector machines, Decision / Model trees, Genetic programming, Fuzzy rule based systems, and nearest neighbor. Few areas where data driven-modeling is used:

- Fuzzy rule are used in data-driven modeling for the prediction of precipitation events and analysis of groundwater model uncertainty (Abebe and Guinot., 2000).
- ANN is also used in data driven-modeling for river stage-discharge relationship (Bhattacharya and Solomatine, 2000).
- Another intelligent controller for water management has been developed by Lobbrecht using ANNs and fuzzy rules (Lobbrecht and Solomatine, 1999).

- M5 model tree is used to forecast discharge of river (Solomatine and Dulal, 2003).
- Many rainfall-runoff processes simulated through data-driven modeling are based upon ANN (Abrahart and See, 2000; Abrahart and See, 2007; Hsu et al. 1995; Hu and Wu. 2007; Minns and Hall, 1996).

5.2.4.3 AGENT-BASED MODELING

Agent-based modeling is another upcoming paradigm of modeling in the hydrology domain. In this type of modeling, software agents dynamically interact with the system, based upon simple rule-based program codes. For example, study of glacier prediction and analysis in Himalayan region uses agents, where many glaciers and behaviors of each of the glaciers are programmed separately as agents. Agent-based modeling is still in an emerging stage and details of implementation of this type of modeling are still under development.

5.2.4.4 THUMB RULES OF MODELING

Modeling represents description of real world, therefore it very important to know the process of how to develop a model and use it. Due to uncertainties in the data input to the modeling system, sometimes output is based upon approximation. It needs to be ensured that the model should be able to produce results in all circumstances. In this regard, thumb rules for modeling are given in figure 39.

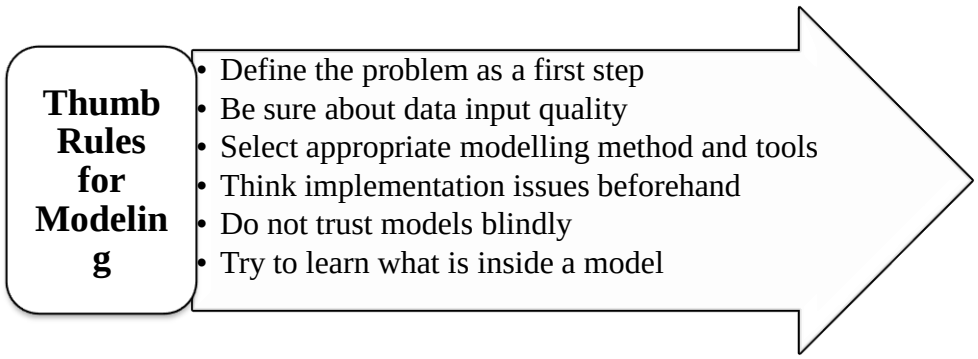


Figure 39: Thumb Rules of Modeling.

To carry out modeling using the above thumb rules, following steps may be taken.

- Collect, prepare and survey the data,
- Identify variables that effect the process,
- Build the model,
- Calibrate selected parameters,
- Evaluate results and data

After carrying out the above steps, apply the model to the system under consideration. ICT always plays an important role in application of modeling.

5.2.5 E-LEARNING IN HYDROLOGY DOMAIN.

IT has become an integral part of everyday life in the present times. Considering the increasing popularity of IT in day-to-day functioning in all spheres of work, it is very effectively being utilized in the field of electronic learning (e-Learning). Domain experts from hydrology are starting to use e-learning systems as a better way of exchanging of ideas. Many learning packages have been developed by national as well as international organizations. These packages cover important elements including various definitions and terminologies used in hydrology. The idea behind the development of such packages is to provide a flexible learning environment and a platform to exchange knowledge in water domain.

Considering the importance of e-Learning, National Institute of Hydrology (NIH) India has developed software named '*Learning Package for Hydrology'*. The menu driven and user friendly package has been developed using HTML. Learners can read the document in a non-sequential way just by clicking on the hyperlinks provided. The idea behind its development was to make available this

package for its use anytime, from anywhere in the world and to choose one's own pace of education.

In this package, an attempt has been made to make the learner understand various phases and processes of movement of water on the earth surface and sub-surface. An exhaustive hyperlinked content can ve accessed from the the default home page of the NIH website.

A user (learner) can directly go to the desired topic using the hyperlinks provided in the webpage's content. After the end of each section (or a topic), a link is available using which the user can navigate back to the contents. The topics covered under the main section are:

- Introduction,
- Precipitation,
- Abstraction from Precipitation,
- Stream flow
- Measurement,
- Runoff,
- Hydrograph,
- Floods,
- Flood Routing and
- Ground Water.

Different sub-topics are also provided under each of the main topics. The learner can acquire the basic knowledge of almost all the important areas of hydrology without referring to any textbook.

5.3 SOFTWARE USE IN DOMAIN

Different software are being used in the hydrology domain to solve domain-specific problems. The use of software in many hydrology related national/international organizations and software companies has been exhaustively studied and has therefore been taken up as a separate section, instead of clubbing it under "Hydroinformatics : information technology (IT) in domain" discussed in the preceding pages.

5.3.1 NATIONAL ORGANIZATIONS

In India, latest software available in hydrology domain are being used by many state and central government organizations, like Central Water Commission, Central Water and Power Research Station, National Institute of Hydrology, Ganga Flood Control Commission, Water Quality Assessment Authority, India Meteorological Department, National Water Development Agency, Indian Institute of Technology, National Institute of Technology, state and central universities and other research institutes. All these organizations and institutes have remarkable contributions in their respective area of operations.

For the sake of this dissertation, the study of software usage at National Institute of Hydrology (NIH) Roorkee and Government of India's Hydrology II project are being discussed as follows.

5.3.1.1 NATIONAL INSTITUTE OF HYDROLOGY

National Institute of Hydrology has an objective to spread the knowledge of Hydrology among engineers and academic organizations.

National Institute of Hydrology has developed in-house a number of software in the field of hydrology, the details of which are given in the table 9 in next page:

Table 9: Software from NIH

Software	Details
Unit Hydrograph Applications for Flood Estimation Package- UHPACKI	The software is used for estimation of small floods in order to effectively and appropriately design small bridges, hydraulic structures, culverts, drainage systems, etc. that can bear the fury of floods. Flood estimation is done on the basis of Unit hydrograph approach.
Flood Estimation for Large Catchments Using Deterministic Approach Package-- FLPACK	The software does the same thing as above but caters to the design needs of larger structures based on estimation of larger floods affecting catchments having an area more than 5000 Sq. Km. Flood estimation is done on the basis of a network model that is developed upon many hydrographs.
Software for Reservoir Analysis - SRA	SRA software is meant for various analyses related to design and management of reservoirs in India. Reservoirs are one of the important components of any water resources development project. Major features of SRA is the use of Sequent Peak algorithm to find reservoir capacity, calculation of rule curve levels with mass balance approach, reservoir operation policy development, etc.
Web Enabled Software for Computation of Evapotranspiration	This web enabled software uses methods like Penman Monteith, Thornthwaite, Hargreaves,Blaney Criddle Turc, Temperature Based etc. for computation of Evapotranspiration

5.3.1.2 GOVERNMENT OF INDIA'S HYDROLOGY PROJECT II

World-bank funded Hydrology Project II in India focused upon the area of Surface Water, Ground Water, Water Quality, Floods and Droughts. Main aim of hydrology project is to promote, apply and extend the efficient use of the Hydrological Information System in thirteen States and eight Central agencies

(HP, 2011). Following software solutions are developed during Hydrology Project II

5.3.1.2.1 DECISION SUPPORT SYSTEM – PLANNING (DSS-P)

DSS-P is used for the planning and management of water resources in the agencies, which are the part of Hydrology Project II in India. The main features of DSS-P are given below in figure 40 and figure 41 shows the list software used in DSS-P.

Figure 40: Features of DSS-P

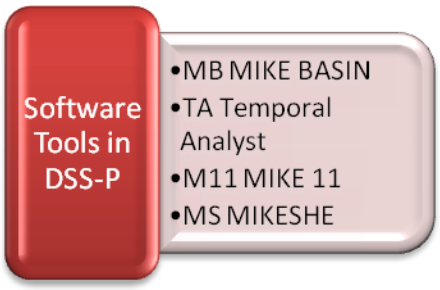


Figure 41: Software Used in DSS-P

Main Capabilities of DSS are Scenario, analysis, data management and reporting.

5.3.1.2.2 HYDROLOGICAL INFORMATION SYSTEM

Hydrological Information System, as shown in figure 42, is a scientific process of collecting reliable and high quality data of hydrological, hydro-meteorological and hydro-geological systems on space and time. It encourages the usage of computerized databases for planning, design and management of better water resources systems in India.

Through Hydrological Information System, the measures of observation, processing and distribution of water resources data can be standardized. Hydrological information in India is primarily provided by various agencies of the central and state government. Focus area are like, proper and timely integration with GIS, design of a enthusiastic hydrological data processing software and standardized activities of ground water and surface water agencies. The software is mainly used for design and analysis of hydrological data.

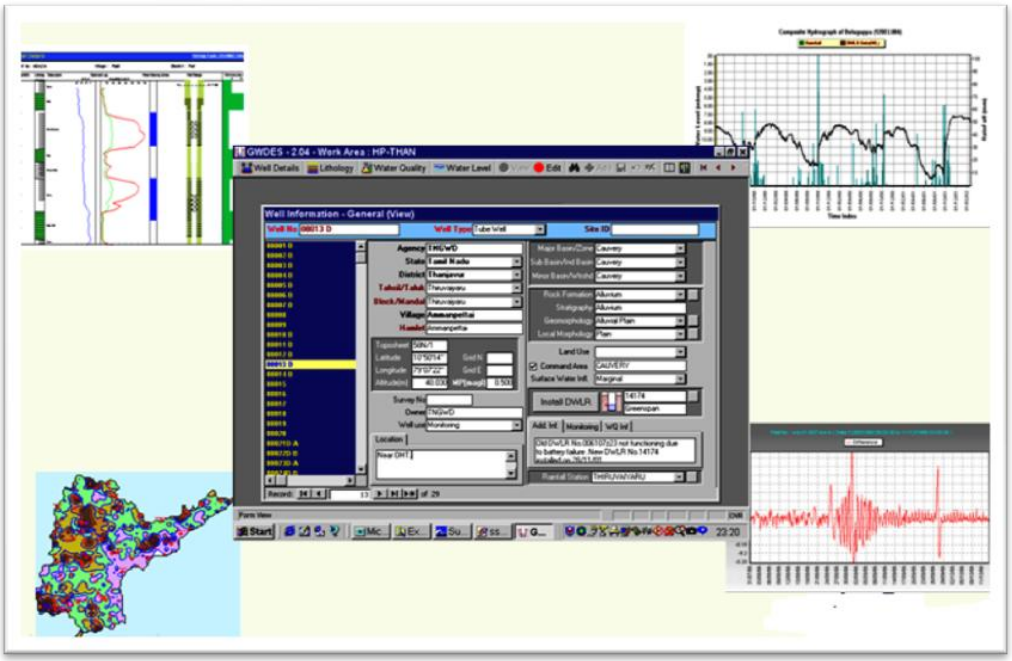


Figure 42: Data Processing & Analysis in HIS

5.3.2 INTERNATIONAL ORGANIZATIONS

Following international organizations are using as well as providing software related to hydrology in public domain. The details of the same are not discussed here, though references to those software have been made under other headings of this thesis.

- The Hydrologic Engineering Center, U.S. Army Corps of Engineers.
- Hydrology and Remote Sensing Laboratory & Agricultural Research Service Department of Agriculture United States.
- Austin University of Texas at Austin.
- United States Geological Survey.

5.3.3 SOFTWARE COMPANIES

Another source of software in hydrology domain is domain specific software companies. They are leaders in their respective fields and are providing smart, valuable software solutions. A few of the leading hydrology related software solution providers are described below.

5.3.3.1 DELTARES

Deltares mostly deals in the field of subsurface, water and infrastructure. The company has contributed remarkably in the field of applied research of water science. Throughout the world, Deltares provides a smart and innovative software solution for water, environment and society. Approximate 60 countries are using the software solutions from Deltares. (Deltares, 2012).

Lot of software solution provided by Deltares are for hydrology and hydraulic systems. Some of the software are Modflow, Sobek, Delft3D and these are developed in a special Hydrosoftware Platform (Vriend, 2011).

Now-a-days flood forecasting, warning and timely response to it may reduce flood related loss. Delft-FEWS is a flood operational forecasting system that

handles real life data and hydrological models in many areas like Rhine basin in Scotland, England and Wales, Netherlands, Switzerland and National Weather Service implemented in United States. Apart from operational forecasting, Delft-FEWS is also used for research development and integration with other systems.

Delft FEWS forecasting system can be integrated with other models, may be global- and local-scale meteorological models like GRIB, GRIB2, or NetCDF-cf and can be linked using database export-import utility. The reports are generated in HTML / XML format and can be distributed on Internet. This forecasting system software allows integration through loose coupling of models. The advantage of loosely coupled approach is widely accepted for integration of simulation modeling, otherwise lots of efforts are required for integration. The loosely coupled and flexible approach is made possible through the use of a simple XML interface (Weerts and Schellekens, 2010). The next figure graphically depicts the software.

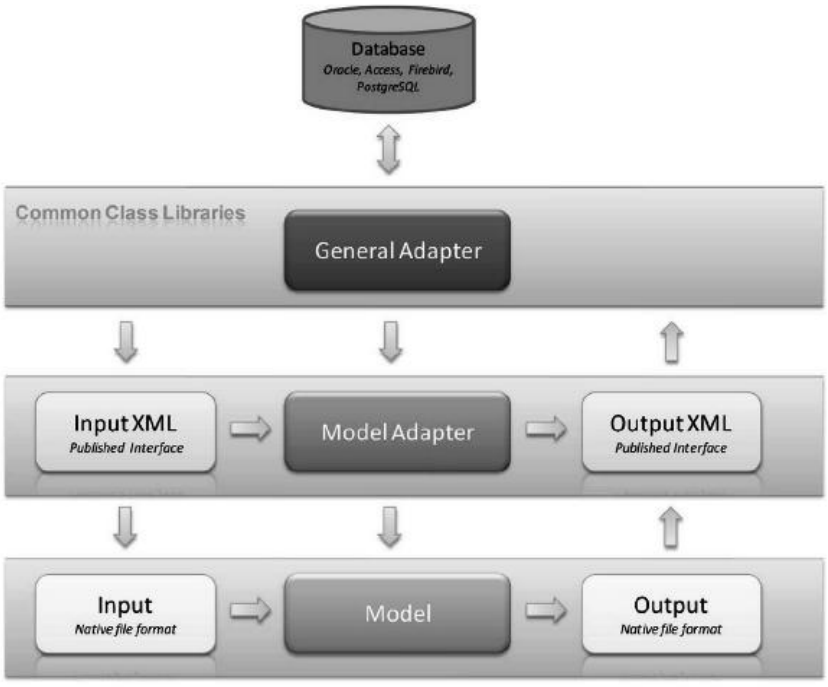


Figure 43: Open communication between Delft-FEWS and models

5.3.3.2 DHI SOFTWARE'S

DHI is leader in software solution and services in hydrology domain. Details of few software solutions are given below:

Mike 11 is a one dimensional river modeling software. It is used for determining the flow conditions, simulating flow and catchment run-off for flood analysis, flood alleviation, flood forecasting etc.

Mike flood software provides a range of 1D and 2D flood simulation engines for modeling. It can be used to model any flood problem, be it related to flooding in rivers, streets, network of drains, land plains, coastal areas, dams or levee breaches, and is therefore helpful for analyzing flood risk for industrial, residential or cultural heritage area, assessing rapid floods, mappings of flood hazards and contingency planning of floods.

MIKE SHE software excellent way of an integrated modeling of groundwater, surface water, evapotranspiration and recharge. It is used in integrated catchment hydrology, conjunctive use of groundwater and surface water, Wetland management and restoration, Floodplain management, Groundwater induced flooding, Groundwater remediation and Environmental river flows,.

5.4.3.3 INNOVYZE

Innovyze provide business analytics software solutions designed to meet the technological desires of water resources, government organizations, wastewater utilities and engineering organizations worldwide. Innovyze have major clients in UK, Australia, East Asian and North American cities.

FloodWorks is a real time simulation forecasting for e-hydrologic and hydraulic conditions of river basins and coastal areas. It can be linked with multiple data source, integration with multiple models in hydrology to predict the flow, velocities, water quality and flood depth. (Body, 2011).

FloodWorks includes three applications as shown figure 44 and figure 45:

Figure 44: Applications in FloodWorks

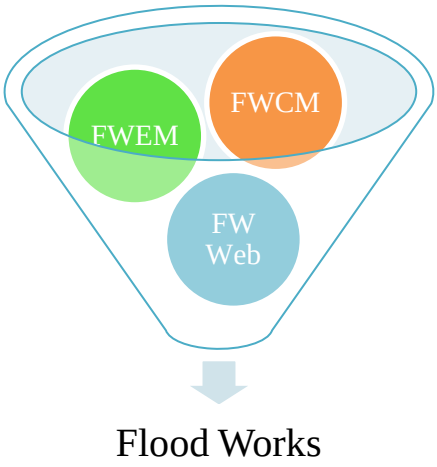


Figure 45 : Flood Works Activities

1D and 2 D simulation, GIS and data management through relational database and output as engineering reports is made available through Infoworks RS, a software solution from Innovyze. It is also used in event based rainfall run off modeling.

5.3.3.4 BENTLEY

Bentley provides the software solutions in Hydrology and Hydraulic domain in the areas of water distribution management and storms. For hydrologic and hydraulics systems requirement, analysis, design, and better management

software solutions from Bentley exist, like WaterCAD, WaterGEMS, SewerCAD, SewerGEMS, and StormCAD.(Bentley,2011).

Navi Mumbai Municipal Corporation (Bentley and NMMC, 2011) has used Bentley software solutions to analyze and manage a water network of more than 600 kilometers in length. Bentely software's solutions helped NMMC to prepare accurate city planning and capital investment plans in humanizing the infrastructure in Navi Mumbai, so that it can fulfill the need of growing population.(Bentley and NMMC, 2011).

Most of the software utilities are created through Bentley's software solutions like SewerCAD, WaterGEMS, HAMMER, Bentley Water, PowerCivil, MicroStation, Bentley Geo Web Publisher and STAAD.Pro

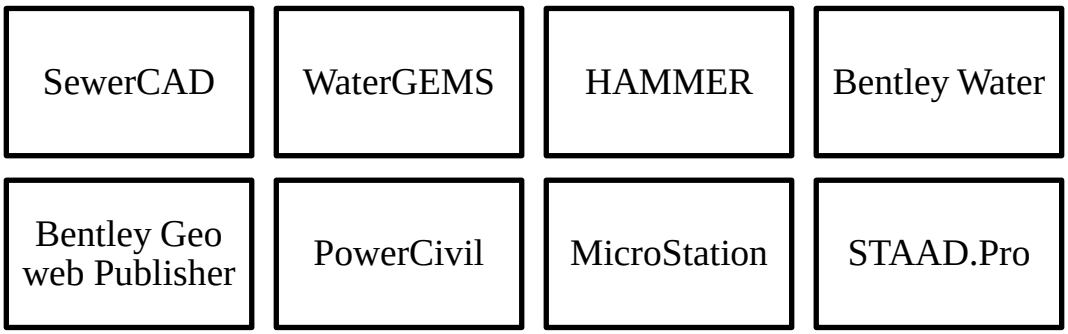


Figure 46: Bentley's Software

Bentely has provided another software solution through a product WaterGEMS which is used by Maharashtra Jeevan Pradhikaran (Bentley and MJP 2011.) for providing 24x7 water supply to more than 140,000 residents of Badlapur city in India. As a result, 427 million liters water are additionally available annually and a saving of 5000000 INR per year. Project has a 30-year lifecycle, that way

savings will be substantial (Bentley and MJP, 2011). Other improvement parameter are given in figure 47 :

Parameter	From	To:
Revenue	91%	96%
Losses	29%	23%
Staff/1000	13.72	10
Complaints/1000 connections	55	5
Annual O&M expense/ 1000 connections	Rs. 34.55 Lakhs	Rs. 29 Lakhs
LPCD	171	135 (Only areas where UG tanks are not leaking)

Figure 47: WaterGEM Contribution in Revenue Performance

Bentley provides software solution in the field of hydrology in India as well as in other countries all over the world.

5.4 STATUS OF DOMAIN-SPECIFIC SOFTWARE IN INDIA

In addition to the studies made that has been described in the preceding pages, an exhaustive survey and interviews with domain experts and hydrologists was conducted to understand more clearly the type of software used in hydrology domain and about the usage of Component-based software engineering approach in designing & development of software in hydrology. This section presents the status of hydrology domain software in India.

5.4.1 ORGANIZATIONS

The domain experts including hydrologists, water managers, scientist, engineers and people from academic research were surveyed and interviewed from the following reputed organizations like:

- Central Water Commission,
- Central Water and Power Research Station Pune, Maharashtra,
- India, National Water Development Agency,
- National Institute of Hydrology,
- Ganga Flood Control Commission Patna India,
- Water Quality Assessment Authority,
- India Meteorological Department,
- Indian Institute of Technology,
- National Institute of Technology,
- State and central universities and other research institutes.

In some of the organizations personal visits were made to meet the experts, where as in others, the responses were collected using web technologies, email and phone calls.

Figure 48 shows the organization participation details in the survey conducted .

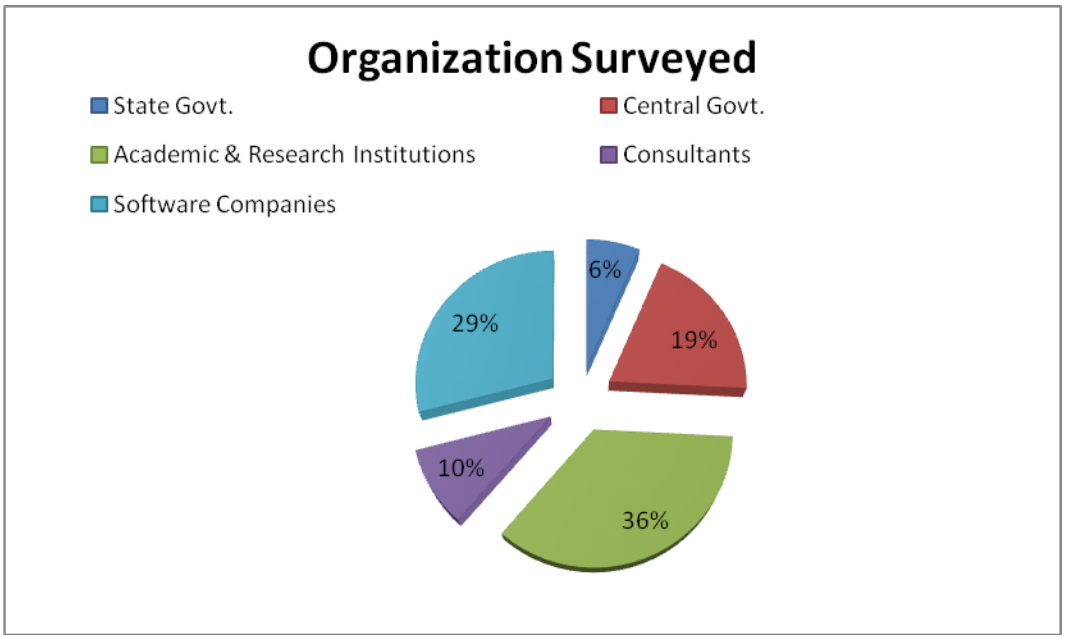


Figure 48: Hydrology domain Software user

Other question asked was about the areas in which software are used in hydrology domain. Major finding is that software related to hydrology are used in either surface water or ground water and in some organizations software are used in both as shown in figure 49.

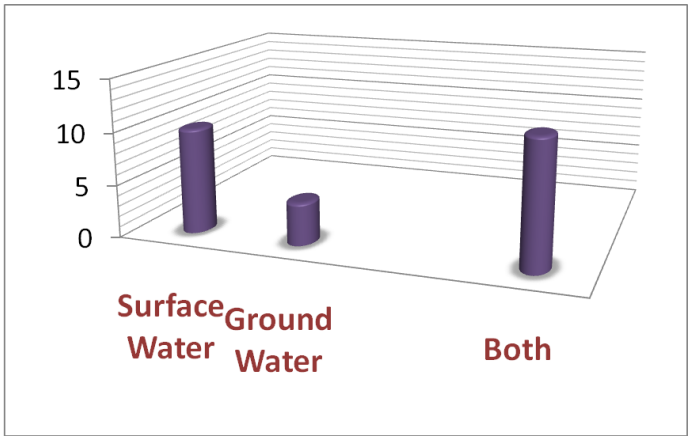


Figure 49: Software Usage

5.4.2 SOFTWARE APPLICATIONS

As discussed in the preceding pages, the software solutions used in hydrology domain in India are either provided by software companies or National and international organizations or in house developed. List of software solution used by organizations and institutes in India is summarized in table 10 and list of in-house developed application given in table 11.

Table 10: Software used in hydrology domain

Sr. No	Software solutions Suggested	Software Source	Purpose
1	WRAP - Water Rights Analysis Package	Austin University of Texas at Austin	Flow naturalization
2	MWSWAT (Map Window Soil & Water Assessment Software solutions)	Texas A & M University (Open source)	Hind-casting of flow data with Rainfall-Runoff modeling
3	HEC-HMS	The Hydrologic Engineering Center, U.S. Army Corps of Engineers	
4	Regression Techniques P-Q and Q-Q (linear, non-linear, multi-linear) Model similar to NAM	In house developed	

Sr. No	Software solutions Suggested	Software Source	Purpose
5	HEC- RECSIM	The Hydrologic Engineering Center, U.S. Army Corps of Engineers	Water resources system modeling
6	WRAP	Austin University of Texas at Austin	River basin modeling
7	WINSRM / SNOWMOD	Agricultural Research Service, Department of Agriculture United States	Snowmelt runoff modeling
8	WINSRM	Agricultural Research Service, United States Department of Agriculture	Glacier melt runoff modeling
9	MWSWAT	Texas A & M University(Open source)	Technique for assessing the potential impact of climate change
10	Unit Hydrograph Applications for Flood Estimation Package UHPACKI	National Institute of hydrology (NIH) India	Tool for development of response function for basins of size less than 5000 km2 which will include determination of T-hour Unit Hydrograph using storm event and concurrent discharge values, Collin's method, Nash model, Clark model
11	HMR52 + In house	The Hydrologic Engineering Center, U.S. Army Corps of Engineers	Tool for storm analysis which includes determination of depth area duration curves and development of storm hyetograph.

Sr. No	Software solutions Suggested	Software Source	Purpose
12	SRM/ Snowmode	Department of Agriculture United States	Tool for snowmelt contribution
13	HEC-RAS	The Hydrologic Engineering Center, U.S. Army Corps of Engineers	Tool for hydraulic routing.
14	In house/ HEC-HMS		For computation of flood hydrograph ▪ Loss models ▪ Transform models ▪ Assessment of base flow ▪ Routing models
15	HEC-HMS		Tool for Kinematic Wave model for urban catchments
16	HEC-HMS		Tool for SCS curve number method for agricultural catchments
17	MIKE SHE	DHI Software's	Integrated catchment & Water Flood Management
18			Irrigation and drought management Analysis
19	FLOOD WATCH	DHI Software's	Client-Server environment integrates real-time data and forecast modeling tools for flood forecasting
20	HYMOS	Delft Hydraulics	Tools for Data processing, analysis and reporting

Table 11: List In House Developed Software Applications

Sr. No	Application Areas
1	Software solutions for graphic representation of best fit distribution and original series with confidence band
2	Software solutions to implement L-moment approach of RFFA analysis
3	Software solutions for USGS method pooled curve method and Analytical methods
4	Software solutions for all empirical formulae
5	Software solutions for estimation of design flood from updated CWC envelope curves
6	Tool for Rational method for both urban and agricultural catchments
7	Software solutions for identification of region of influence (ROI) of the ungauged basins
8	Software solutions for data mean, SD, skewness, kurtosis and detection of outliers.
9	Software solutions for estimation techniques based upon four parameter Method
10	Software solutions for IDF curve analysis.
11	Software solutions for determination of parameters of channel routing
12	Software solutions for integrating GLOF with the intermediate catchments runoff.
13	Software solutions for Synthetic Flow Generation
14	Software solutions for Data validation and gap filling

Many software solutions are used in hydrology domains by State / Central Government organizations, academic and research institutes. Sources of software's in hydrology are: Software companies, Research institutes and In house development. Next chapter explores the possibility of software reusability in domain through Component-based Software Engineering and design a framework for the same.

CHAPTER 6

DOMAIN SPECIFIC CBSE FRAMEWORK

This chapter provides overview of existing software in the identified domain of hydrology and proposes a domain specific Component-based software development framework. Software component repository along with one application also design and developed through framework.

6.1 SOFTWARE REUSABILITY THROUGH CBSE IN DOMAIN

As discussed in Chapter 5, three different sources of software being used in the hydrology domain are:-

1. Leading software companies like Deltares, DHI Software's, Innovyze and Bentley. These companies are providing the commercial software for hydrology domain.

2. National and international organizations providing software in hydrology in public domain, like IIT Kharagpur, NIH and IIT Roorkee in India and Hydrologic Engineering Center, U.S. Army Corps of Engineers, Hydrology and Remote Sensing Laboratory & Agricultural Research Service Department of Agriculture United States, Austin University of Texas at Austin, United States Geological Survey at the international level.

3. In-house software developed or a module integrated with public domain/commercially available software.

United States Geological Survey has categorized application software being used in hydrology into four categories corresponding to the four areas of hydrology,

namely water quality and hydro chemistry, surface water, ground water, statistical analysis & graphics. The survey results obtained during analysis of software usage in the Indian context as discussed in Chapter 5 have also been taken into consideration in identifying these four categories.

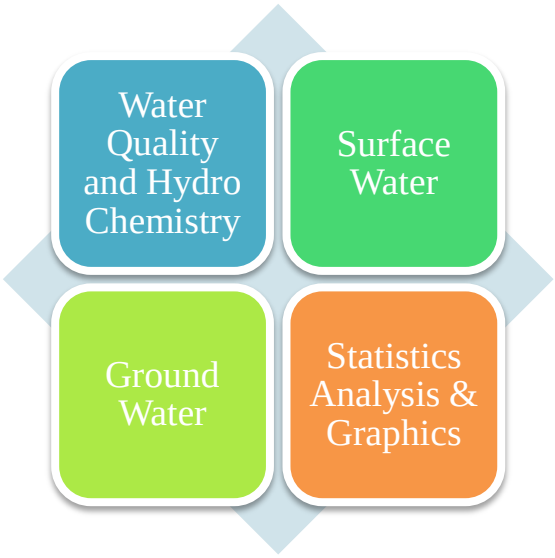


Figure 50: Software categorization in hydrology domain

The scope of this research work has been narrowed down to focus only on the software being used in the surface water category of hydrology. Surface water deals with water above the earth's surface, for instance water found in rain, drizzle, snow, hail, lakes, ponds, rivers, floods and oceans. Approximately 40 software dealing with surface water have been studied for the purpose of this research work. To find out the possibility of software reusability using CBSE, these 40 software have been further analyzed on the basis of following parameters as shown in figure 51.

Figure 51: Parameters for Software analysis

To explore the concept of reusability in hydrology's surface water domain-specific software, experimental setup was conceptualized and implemented using many computers, each with an operating system (for instance, DOS, different versions of Windows, Unix, Linux, Solaris, etc.) and specifications (like Dot Net, JDK, different flavors of FORTRAN compiler from 16-bit to Visual FOTRAN compiler, etc.) confirming to the requirements of the software-to-be-analyzed. For analysis purposes, trial versions of commercial application software, public domain application software and application software developed in-house were used.

Some of the application software under consideration of this research work, like the commercials ones are proprietary. In such cases, development methodologies are not disclosed by the concerned companies.

Considering the market competition, state of application development in the ICT industry and reputation of the software vendors providing software solutions in hydrology, reasonable assumptions were made that the software companies may have implemented best development practices including reusability in developing & supporting these application software.

An analysis of the application software under consideration of this research work suggests use of legacy applications that have been developed using procedure based languages like FORTRAN where reusability of software components may not be possible, as understood in the present-day context.

The finding of such an experiment was that there was a very limited role of software reusability through Component-based software engineering in hydrology's surface water domain-specific software under consideration. Partial use of software reusability was found in Rainfall Runoff Software Library (Perrauda et al. 2003) and Open Modeling Interface (Moore 2010a).

6.1.1 THE RAINFALL RUNOFF SOFTWARE LIBRARY

Partial usage of reusability of software has been found in the Rainfall Runoff Software Library developed by Cooperative Research Centre for Catchments Hydrology, Canberra Australia to satisfy the need of software end user community in hydrology domain. They tried to apply the concept of software reusability in Rainfall Runoff Software Library by creating software library of rainfall runoff models like AWBM, TankModel, SimHyd and Sacramento. Figure 52 shows the model selection in RRL Library. It is designed in such a way that there is a separate feature in the form of a toolkit for implementation of each of the tasks, like Input / Output, modeling, calibration and visualization. This library is implemented in .NET environment (Perrauda et al. 2003). In this software library OOAD have been used for software reusability for limited number of hydrology's rainfall runoff models.

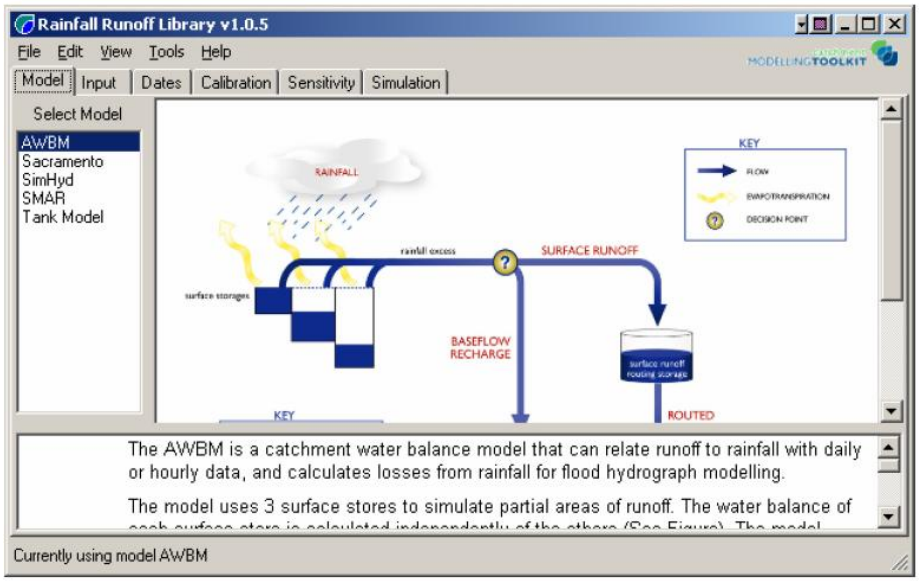


Figure 52: Model Selection in RRL Library

6.1.2 OPEN MODELING INTERFACE

Another instance of software reusability in hydrology's surface water domain was found in the integration of modeling. Traditionally, integration was done by manually arranging data output of one model as an input into another model. Open Modeling Interface has changed this approach. OpenMI standard provide interface definitions to computational models of hydrology's surface water domain. In other words, software solution of hydrology model is developed on the basis of the interface specification provided by OpenMI. The interface specifications also take care of integration issues. (Moore, 2010b).

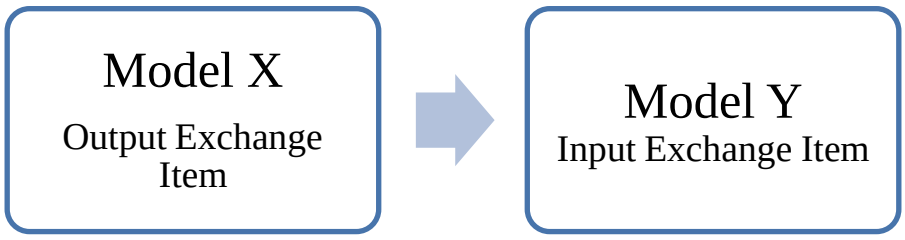


Figure 53: Input Output Exchange within Model

Since 2002, the open modeling interface (OPENMI) specifications and standards are developed with the support of HarmonIT consortium. Members of HarmonIT consortium include DHI, Delft Hydraulics, CEH, National Technical University of Athens, etc.

The OpenMI interface specifications are available both in Java and C#. To work on OpenMI, minimum requirement of hardware as well as software are to be understood. For instance, .NET framework is required to run OpenMI in C#, whereas JDK is required to run OpenMI in Java. The latest version of OpenMI implementation in Java and C# are available from sourceforge web site. Wherever model has been developed using OpenMI specifications, it has been found that partial software reusability is available. In other words, models developed and integrated using OpenMI specifications offer instances of reusability whereas the

ones developed and integrated without OpenMI specifications do not offer instances of reusability.

6.2 IDENTIFICATION OF REUSABLE SOFTWARE COMPONENTS

The identification of reusable software components has been an important activity in the experiment conducted for this research work. For this purpose, first the usability and applicability of the 40 software considered for this research work were explored. As a result of such an exercise, major software applications used in surface water were categorized into areas of applications, like data-integration and management, rainfall runoff model, river flow and flood forecasting, models and work flow management, and statistical analysis. Categorization of surface water software on the basis of applications area is given in figure 54.

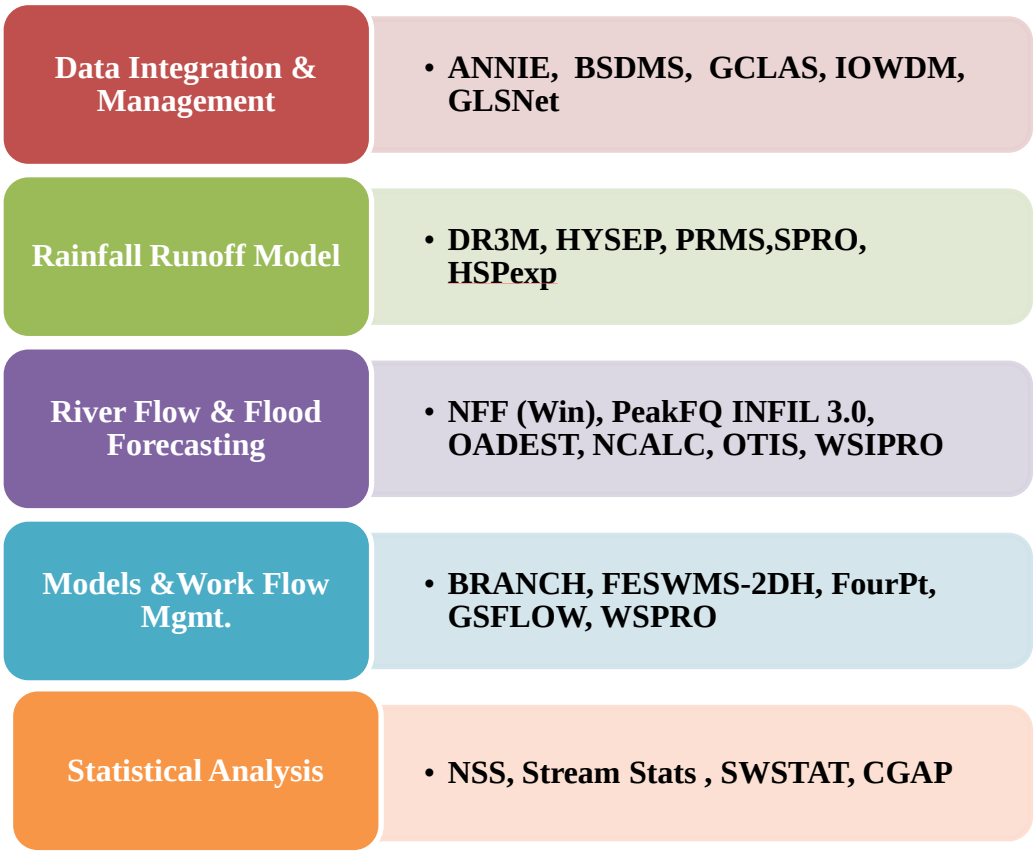


Figure 54: Application Wise Categorization of Surface Water Software.

Further to the categorization exercise, two application areas were chosen in the experiment setup for this research work to identify the possibility of reusable software components through components based software engineering approach. The first application area is data integration and management and the second one is rain fall run off process.

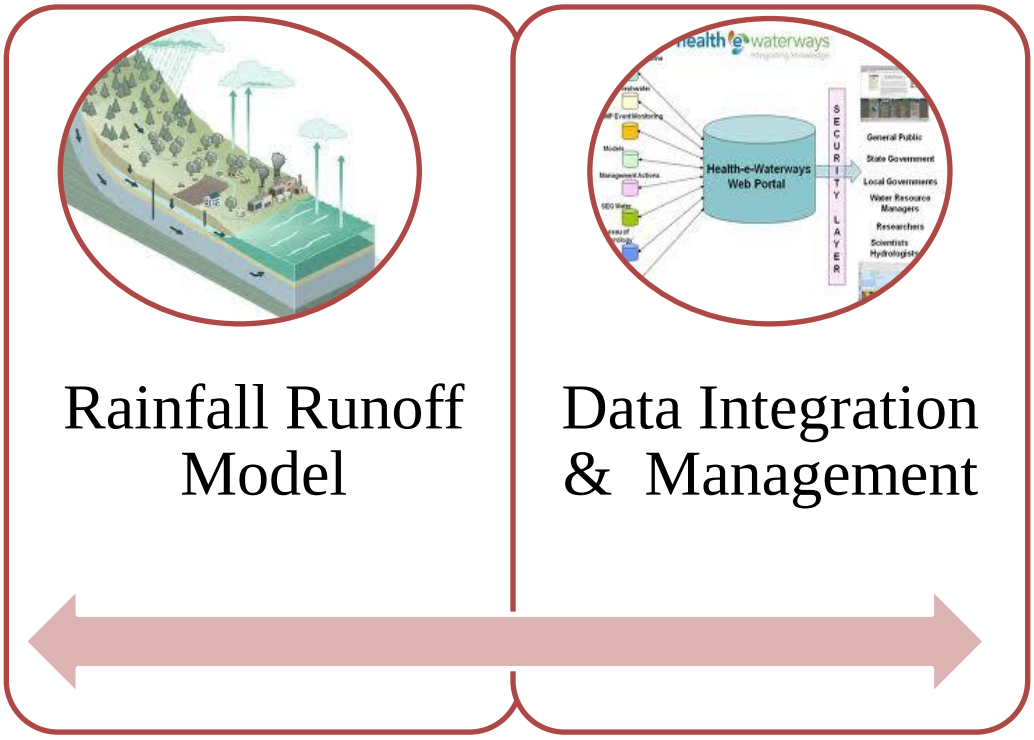


Figure 55: Software Application Area

After understanding the functionality of the software in these two application areas, common features in each of these applications were identified. It was accordingly concluded that such common features are the best candidates for reusable software components in each of the two application areas considered for the purpose of experimentation. A list of such reusable software components under each of the application area is given in next page.

Possible reusable software components Rain Fall Run off Models are as follows:

- Software Components different Rainfall Runoff Model
  - AWBM
  - Sacramento
  - SimHyd
  - SMAR
  - Tank Model etc.
- In Designing of User Interface
- Data Export / Data Import
- Calibration
- Hydrograph
- Water Simulation
- Hydrological Model
- Water Reservoir Model
- Models Optimization
- Google Maps Integration
- River Model
- Surface Water Modeling
- Hyetograph
- Infiltration Estimation
- Rain Fall depth Methods
- Anomalies Removel in Rain fall (Missing Rail Fall record, Inconsistency in data recording)
- Snow Calibration
- Precipitation – run off process

Possible Reusable software Components for Data Management and Integration is as follows:

- Export Data from Different source,
- Data Cleaning,
- Data Conversion,
- Data Noise Removal
- Specific Query
- General purpose Query
- Reporting
- Software Components designing user Interface
- Inflow data exchange
- Out Flow Data Exchange

These are the possible areas where reusable software components may be designed and may be used in multiple applications, so as to avoid developing software applications from scratch.

6.3 NEED OF DOMAIN SPECIFIC COMPONENT BASED SOFTWARE ENGINEERING (CBSE) FRAMEWORK

The discipline of software engineering provides state-of-art development practices for software development. However, the Component Based Software Engineering (CBSE) approach has an edge over other ones, which is briefly described as below.

In the traditional Systems Development Lifecycle (SDLC) model, the application software is developed from scratch, whereas in the Component Based Software Engineering (CBSE) approach, application software is developed using pre-build software components (Storey and Sugumaran, 2003;Waguespack and Schiano, 2004).

Component Based Software Engineering (CBSE) based approach is different from Object-oriented and structured approach (Sharp and Ryan, 2005).

In the traditional Systems Development Lifecycle (SDLC) model, the approach is more structured wherein applications are developed, like a waterfall, in stages - planning, requirement analysis, design, coding, testing and implementation, and maintenance. The approach is very systematic and logical to develop a robust software application. However, the main difficulty in Systems Development Lifecycle (SDLC) approach is the perceived difficult to modify it easily. It is not easy to effectively gather exact change requirements from the user and as a consequence the maintenance cost become very high (Hoffer et al., 2005).

In the structured approach, sometimes there is a very high level of dependency within a block of code. Modification in one part of the program code shoddily affects other parts of program code, whereas in Component Based Software Engineering (CBSE), software components are developed as a standalone project. As it is deployed and tested in isolation, therefore it does not affect the working of other software components (Sparling, 2000; Waguespack and Schiano, 2004).

Moreover, in Component Based Software Engineering (CBSE), replacing & upgrading of the software components have very little effect on other software components (Hopkins, 2000).

However, in Component-based Software Engineering (CBSE) approach, the selection of appropriate component is very important, that is the designer and the

developer need to be aware of the possible software components required for the job (Waguespack and Schiano, 2004).

Object oriented analysis and design (OOAD) approach made the system essentials more usable, increased system quality and usefulness of analysis & design phase. OOAD is built on the encapsulation of data and processes into objects. These objects represent real world entities. There are many different phases in OOAD, but the general phases are analysis, design and implementation (Hoffer et al., 2005; Hopkins,2000).

The Foundation of Component-based Software Engineering is OOAD (Hopkins 2000; Waguespack and Schiano, 2004).

Use of web services in web based solutions for ecommerce or domain specific applications is also the part of Component-based Software Engineering (Yang 2003).

There is always a argue in the academic world of whether web services are software components or not. This research work agrees with the proponents of web services as software components.

As mentioned earlier, one of the main objectives of Component-based Software Engineering (CBSE) is to develop application from pre-existed software components. Many organizations use software components which are developed by third party also (Vitharana,2003).

The studies during this research work has led to a general observation that though lot of work have been done in the area of Component-based Software Engineering (CBSE), yet there is much scope for further work.

There are many general purpose software components frameworks like EJB (Hauge and Sørensen.,2009;Emmerich,and Kaveh 2002).NET (Kum and Kin,

2006), CORBA (Schmidt, 2000) as well as FRACTAL (Bruneton at el., 2004) that are used widely in the software industry.

There is very little domain specific software component based development like AutoSar (Heinecke, 2004), SaveCCM (Hansson et al., 2004) Pin (Hissam et al., 2005), etc. There is a need to work in this area.

This research work proposes promotion of Component-based Software Engineering (CBSE) approach in a domain. Component framework is a set of frequently implemented protocols to connect software components and enforce some of the policy set in the framework. The policies define the mechanism used by the framework(Szyperski 2011).

Before developing a framework, a model application in the hydrology domain is discussed in the next section.

6.4 MODEL APPLICATION

In general, a model application in hydrology domain consists of a user interface and a Shell. The shell is where the calculations will take place (Moore 2010b). Data comes through user interface and it goes to shell. The user interface generates input files for the shell. The user can run the model simulation e.g. by pressing a button in the user interface, which will deploy the shell . The shell will read the input files and perform processing and generate the result which is written in to a output files. Output may be the the form of text file, excel or graph.

In most of the application in hydrology sector time series data is used. Figure 56 depict the model application in hydrology domain

Major software used in surface water are data-integration and management, rainfall runoff model, river flow and flood forecasting, models and work flow

management, and statistical analysis.

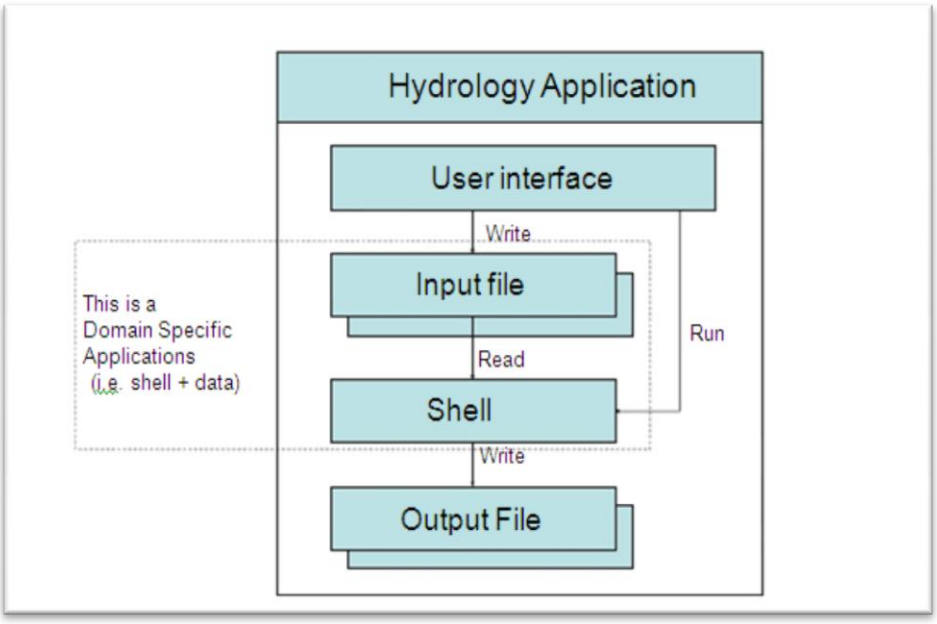


Figure 56 : Model Application in Hydrology Domain

6.5 DOMAIN SPECIFIC CBSE FRAMEWORK

This research work proposes a domain specific Component-based Software Engineering (CBSE) framework that includes the development of software components as well as development of applications from these components. Organizations using the framework may use in-house developed software component or software components purchased/arranged from third party, reuse already purchased or already developed/arranged software components and even Open Source Software (OSS) component.

Now days software companies are widely adopted and produced OSS components to build of software applications through OSS components(Hauge and Sørensen, 2009).

For this purpose, theoretical CBSE framework proposed by Sharp & Ryan based upon design science principals (Sharp and Ryan 2010) has been modified and is used as a base for further work to make domain specific framework.

This research work proposes component-based Software Engineering (CBSE) approach for the development of software components based upon the need of a domain. The software components so designed and developed are then to be used and reused in domain's application. Figure 57 depict the domain specific CBSE framework.

The proposed theoretical domain specific CBSE framework has two parts / activities:

1. **Software Components Development**

2. **Application Development through software components**

Both the activities are iterative in nature and are focused on the needs of the domain. The framework so proposed is unique due to the iterative process, wherein continuous feedback is received from the end users at every stage and gets implemented as per the need of end users.

The approach being taken for developing the CBSE framework is to develop hydrology applications using reusable domain specific software components as well as general purpose software components.

With reference to the figure 57, reusable software components may be identified in user interface, shell, input and output. Shell consists of models, mathematical equations, domain specific logic to solve the problems of hydrology (Moore,2010a).

General purpose software components may be used in creating user interfaces, authentication processes, payment process, etc., whereas domain specific software

components are based upon domain specific functionalities and are used only in applications in that domain.

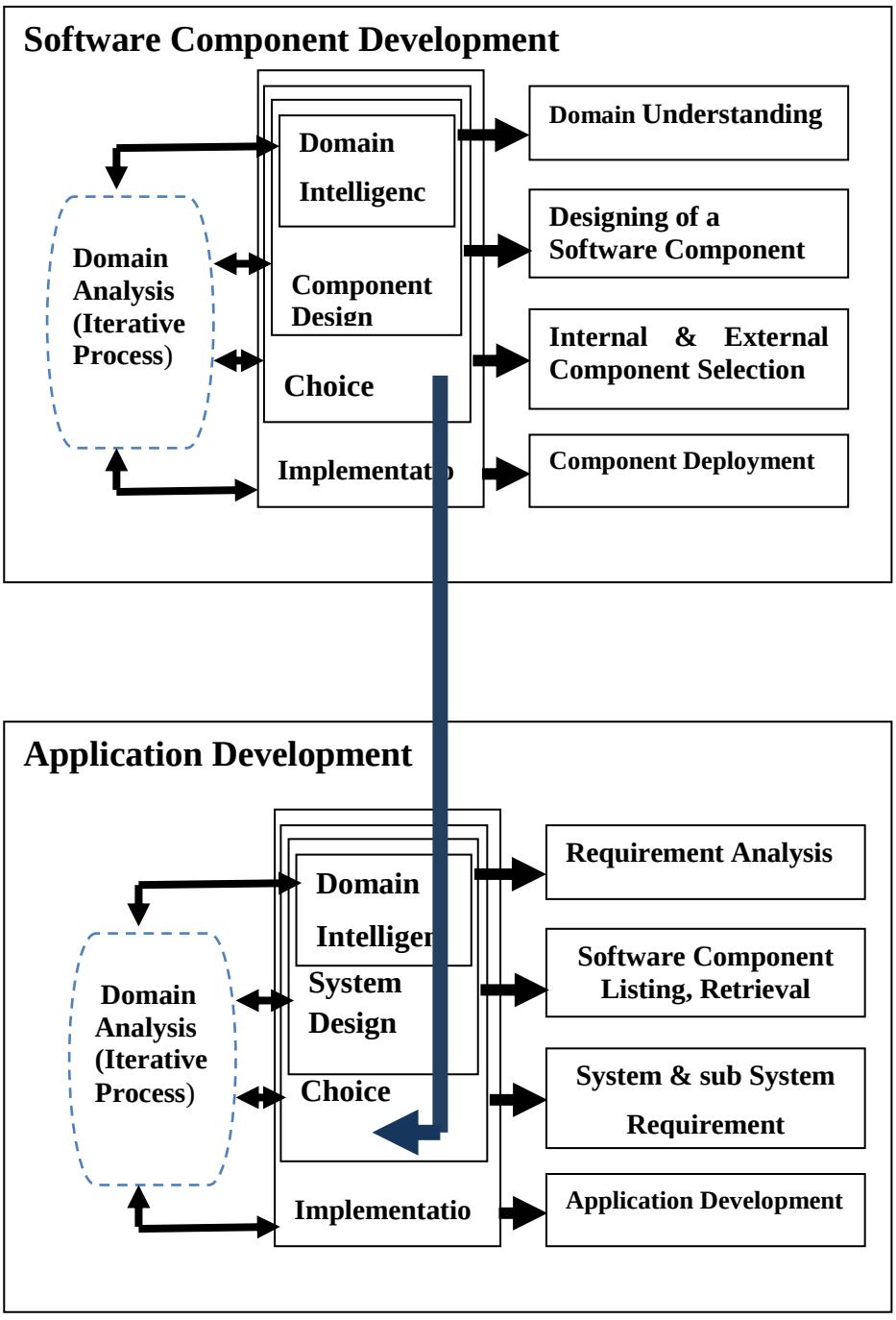


Figure 57 : Domain Specific CBSE Framework

6.5.1 SOFTWARE COMPONENT DEVELOPMENT

As depicted in the figure 57, software components development process has following steps - domain intelligence, component design, choice and implementation. Domain intelligence deals with identification of domain problem that needs to be solved. Problem domain for which software components are to be developed have potential market also (Hopkins,2000). The second step, component design deals with inventing, developing and finding possible future course of action. In this step software component architecture is determined. Design factor for software component architecture are reusability, functional areas, accurate encapsulation and well defined interface (Kim and Bae, 2007). The third step is choice where internal and external functionality are selected. The fourth and final step is implementation where actual software components are created through identified programming paradigm / language (Jain and Vitharana , 2003).

6.5.2 COMPONENT BASED APPLICATION DEVELOPMENT

Again, as depicted in the figure 57, components based application development process has following steps - domain intelligence, system design, choice and implementation. The first step, domain intelligence provides clear understanding of user needs. One of the common reasons for failure of a software application is lack of clear understanding of requirement of the users. Requirement analysis phase gets improved through Component-based Software Engineering (CBSE) approach.

The second step, system design is a major step wherein appropriate component model is selected in order to effectively select, arrange or develop software components and identify their reuse in the software applications. Some of users/developers are of the view that there are not sufficient components available

to fulfill the increasing needs of the end users (Seker and Tanik 2004). The third step is choice where software components are selected and retrieved based upon the previous phase. The fourth and final step is implementation wherein integration of software components takes place to build the desired component based software applications (Jain and Vitharana , 2003; Vitharana and Zahedi, 2003).

6.5.3 SOFTWARE COMPONENT DESIGN ISSUES

The domain specific components model of this research work is derived from the general purpose software components model (Lau and Ornaghi, 2005). Two basic entities exist in this model computational shell and a well-defined interface. Computational shell deals with the computation part like implementation of model logic and through the interface interaction is made with the computational shell. Computational shell is composed of classes with well defined implementation of methods within. There are two design types for designing software components:-

- Atomic design
- Composite design

6.5.3.1 ATOMIC DESIGN

Atomic design consists of a computation unit known as computational shell with a well defined interface through which component interact with others as shown in figure 58. It is the simplest design, first of all, atomic components are designed .

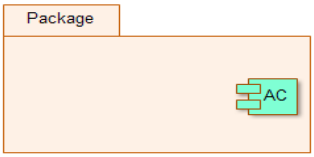


Figure 58: Atomic Components

6.5.3.2 COMPOSITE DESIGN

Using composite design, components are created with the set of atomic components or composite components as shown in figure 59. Components are combined through interfaces and composite interface is created using communication of composite components. `

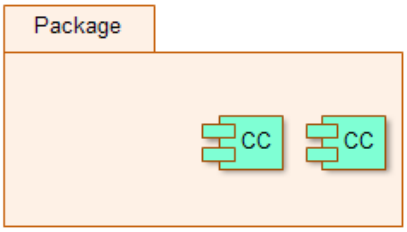


Figure 59 : Composite components

Composition of components is either unidirectional or bidirectional, as per the need the system design. Shown in figure 60 and 61. In unidirectional composition, a software component requiring data input requests the data from other software component providing the output as shown in fig x. Sometimes composition of software components requires exchange of data in both directions. For example, a river model requires input from drainage model and in some cases, the drainage model also requires data from river model.

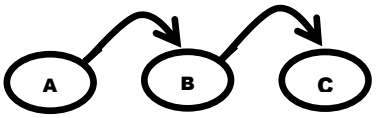


Figure 60: Unidirectional Composition

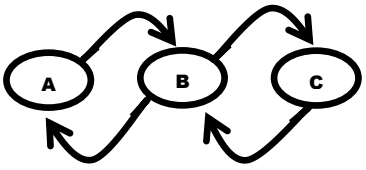


Figure 61: Bidirectional composition

6.6 DATA SPECIFICATIONS – WHEN, WHAT, HOW?

- Metadata specify exchange of data sets within software components. In most of the cases, output of a component may not be fit as an input of other components. Thus data exchange specifications should be based upon

- **What should be exchanged?**

  Quantity: For example, water level (meter) and amount of flow (cubic meter hour)

  Quality: For example, Land use (Agriculture, Housing, Industry) Soil type (Peat, Sand, Clay)

- **Where are the values defined?**

  For example, these may be related to shape and area, may be horizontal / vertical, etc.

- **When the values are needed and its frequency?**

  For example, duration & time interval

- **How the value are to be transferred?**

  Format should be same, so that data can be transferred easily from one component to another.

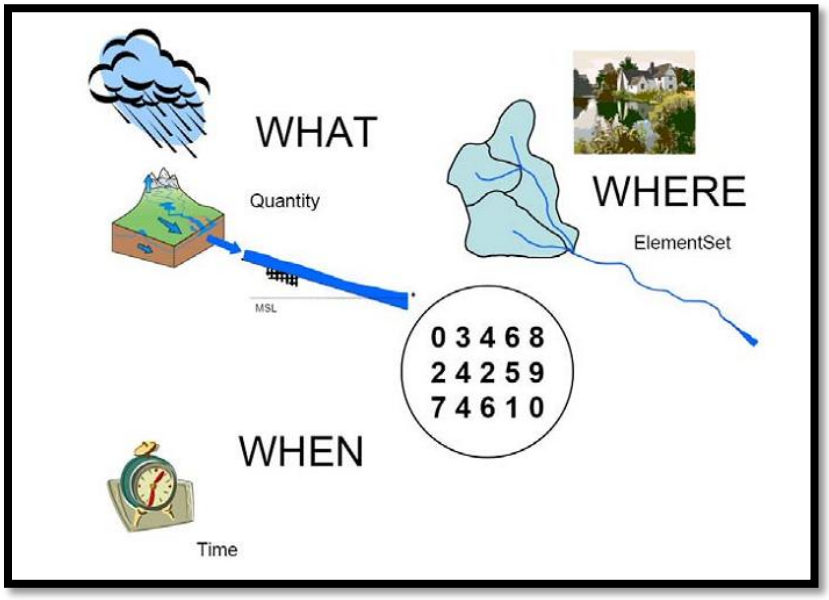


Figure 62: Datasets value

6.7 SOFTWARE COMPONENT REPOSITORY

During the course of this research work, a need was felt to have a repository that should contain domain specific software components, general purpose software components, Open Source Software (OSS) components and Commercially off-the-shelf (COTS) to be made available to water scientists, engineers, hydrologists, researchers and water managers to refer to, use and reuse them whenever there is a need of designing, developing, augmenting or modifying software applications in the domain of hydrology. The concepts of software reusability are easily applied through the use of software component repositories (Luqi and Zhang, 2007). Software components repository provides a platform for successful deployment of several software components in software applications to facilitate producing reliable, secure, affordable software systems.

To fulfill this need, a software component repository dedicated for developing software applications in hydrology domain through CBSE approach was designed and implemented, as discussed in the next section.

6.8 IMPLEMENTATION OF SOFTWARE REPOSITORY

The software component repository as desired in the previous section has been created for the purpose of this research work. Following are some of its salient features:

- It is designed as a web-based one, so that it can be accessible 24x7 - anywhere, anytime through Internet.
- Following open source software has been used:
    - **IDE:** Eclipse **Development language:** PHP **Database:** MySql
- **Why Open Source?** Open source software provides the cutting-edge solution platform. It offers lower development costs with minimal support

and license fee when compared to proprietary software (Androutsellis-Theotokis et al., 2010). Open Source software are community driven and a large number of intelligent and generous force of developers is available online. From a security point of view, open source software comes with source code and are thus more secure as compared to proprietary software (Payne and Christian 2002). As one of the leading player in open-source, IBM offers many quality solutions with the Linux operating systems. Another company, RedHat also in distribution and support services in Linux platform.

- In order to provide authenticated and controlled access to authorized users to the software component repository, the default page of the web-based repository asks for username with passwords and has been so designed to accept only valid registered users only. Figure 63 shows the login screen to the software repository.

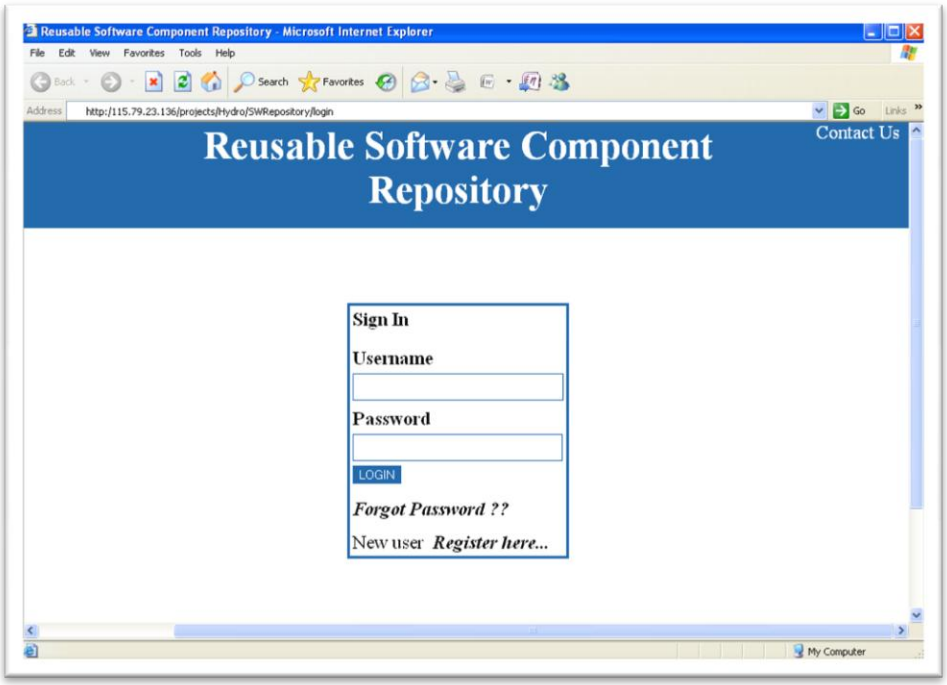


Figure 63: Login page of Software Component Repository

- For new users desiring to use software repository, an option *register here…* has been provided, by clicking which they can get themselves registered and receive username and password through their respective email-id provided during the registration and after the details of the user are verified by a procedure standardized for the purpose. A controlled access has been designed for the purpose of the research work.

- Another option, "Forget password?" has also been provided to the existing valid users who have forgotten the password. When the users click this option, they are asked certain questions, which when answered and submitted are verified and validated with the user's profile data. After successful verification, the password is sent to the registered email-id of the user.

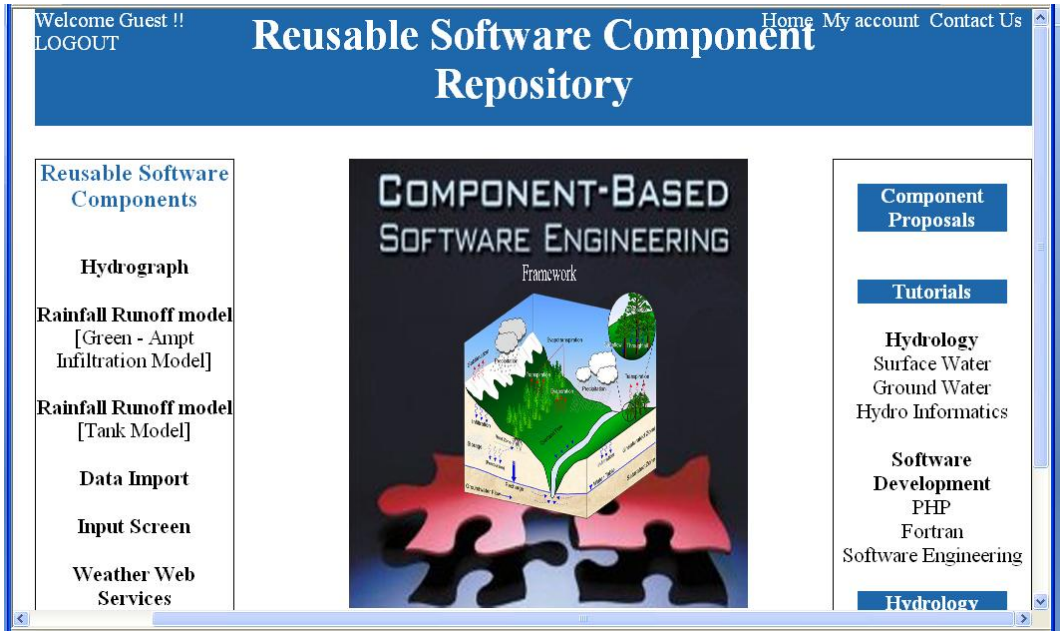


Figure 64: Domain specific software component repository

- After successful login, the user is led to the main page of software component repository web application shown in figure 64. This page provides list of repository's domain specific software components that are

displayed on web-page's left-hand column available and are for use/reuse. A right hand side column on the screen provides option of component proposals, tutorials and links of software in public domain. Top of this screen provide the option to logout, modification of account and contact us.

- Figure 65 displayed domain specific software components are developed through Domain Specific CBSE framework proposed in section 6.5.

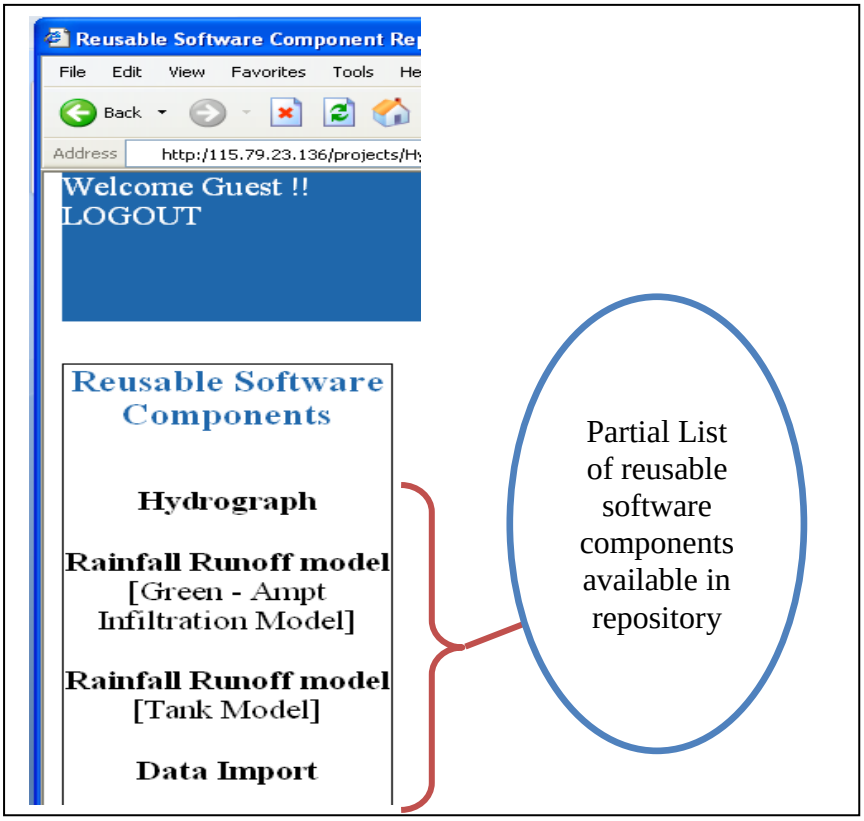


Figure 65: Software Components in Repository

- The following software components developed and uploaded in the web based software repository.

- for generation of hydrograph,
- for making mathematical calculations & logical analysis of two types of rainfall runoff model, viz., Tank model and Green ampt infiltration model,
- for providing data import utility,
- for generating an input form, and
- for a web service to present weather report.

- When any of the displayed software components are clicked, the middle column displays a table containing the summary of that particular software components, type of license like public or GNU, available versions of that software component, and bug summary (if any). Figure 66 and 67 displays the tables for hydrograph software component and for rail fall runoff model – Green Ampt Infiltration Model, respectively.

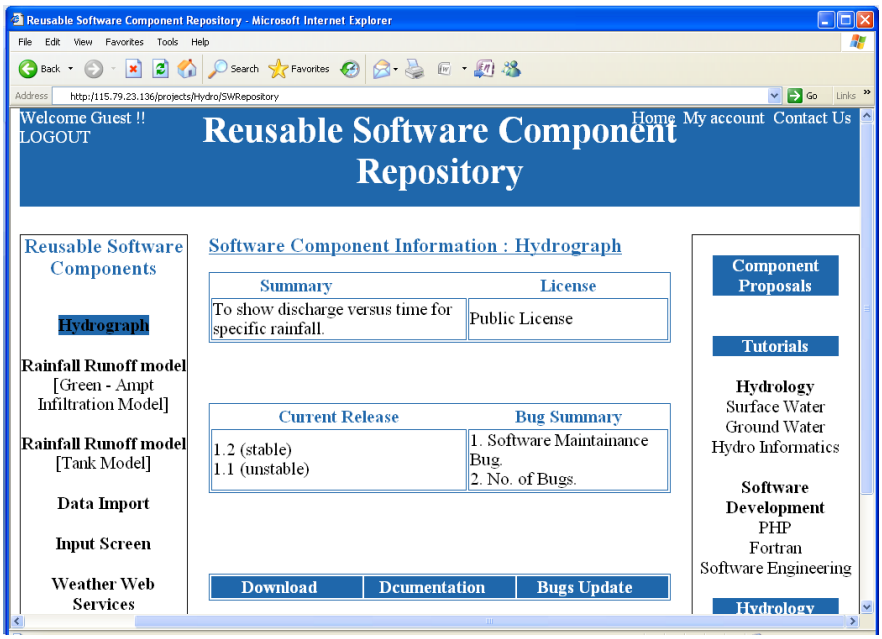


Figure 66: Software component information – Hydrograph

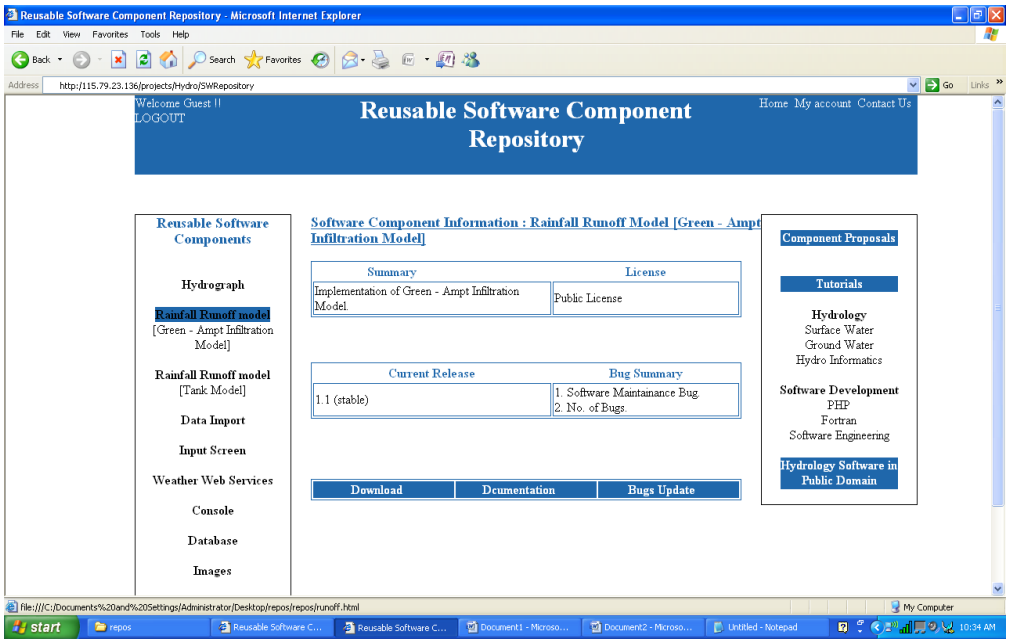


Figure 67 : Software component information – Rain Fall Runoff

- A bar at the bottom of the software repository's webpage as shown in figure 68 has a Download option, which when clicked provides the facility to download available software components on the local client machine, whose user has already been authorized when he/she logged in to browse the repository. The authentic user of the software component repository can use/reuse any of the software components for developing software applications.

- Another button on the same bar at the bottom of the software repository's webpage provides Documentation option, which when clicked provides full documentation support containing details, like specifications etc. for each of the software components.

- Yet another button on the same bar at the bottom of the software repository's webpage provides Bug Update option, which clicked provides the facility to report bugs in any of the software component observed during its implementation. A value added Web 2.0 feature has been provided along with the facility where users can suggest possible solutions through different threads initiated/running on the blog.

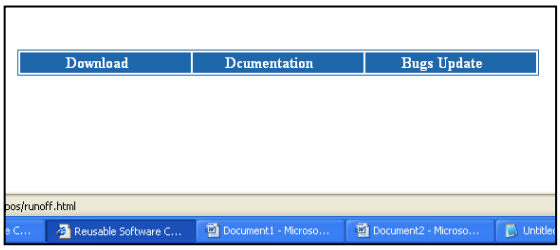


Figure 68: Different Buttons in Software Repository

- The bottom bar with Download, Documentation and Bug Update options is displayed only when a particular software component is clicked for its details.

- As mentioned earlier, the right-hand column has an option to submit proposals of new/modified components for the repository.

- The right-hand column has also a tutorial on two areas – one on hydrology related topics, like Surface Water, Ground Water, Hydro Informatics, and the other tutorial is related to software development technologies, like PHP, FORTRAN and Software Engineering, either of which when clicked provides detailed learning to the user on that particular technology. The topics are suggestive, are not limited to the ones displayed on the web-page and can be expanded / removed as per current requirements.

- The right-hand column also has a link of public domain software in hydrology, which when clicked provides the list of such software for

helping the hydrology community to keep themselves updated on latest related develop as shown in figure 69.

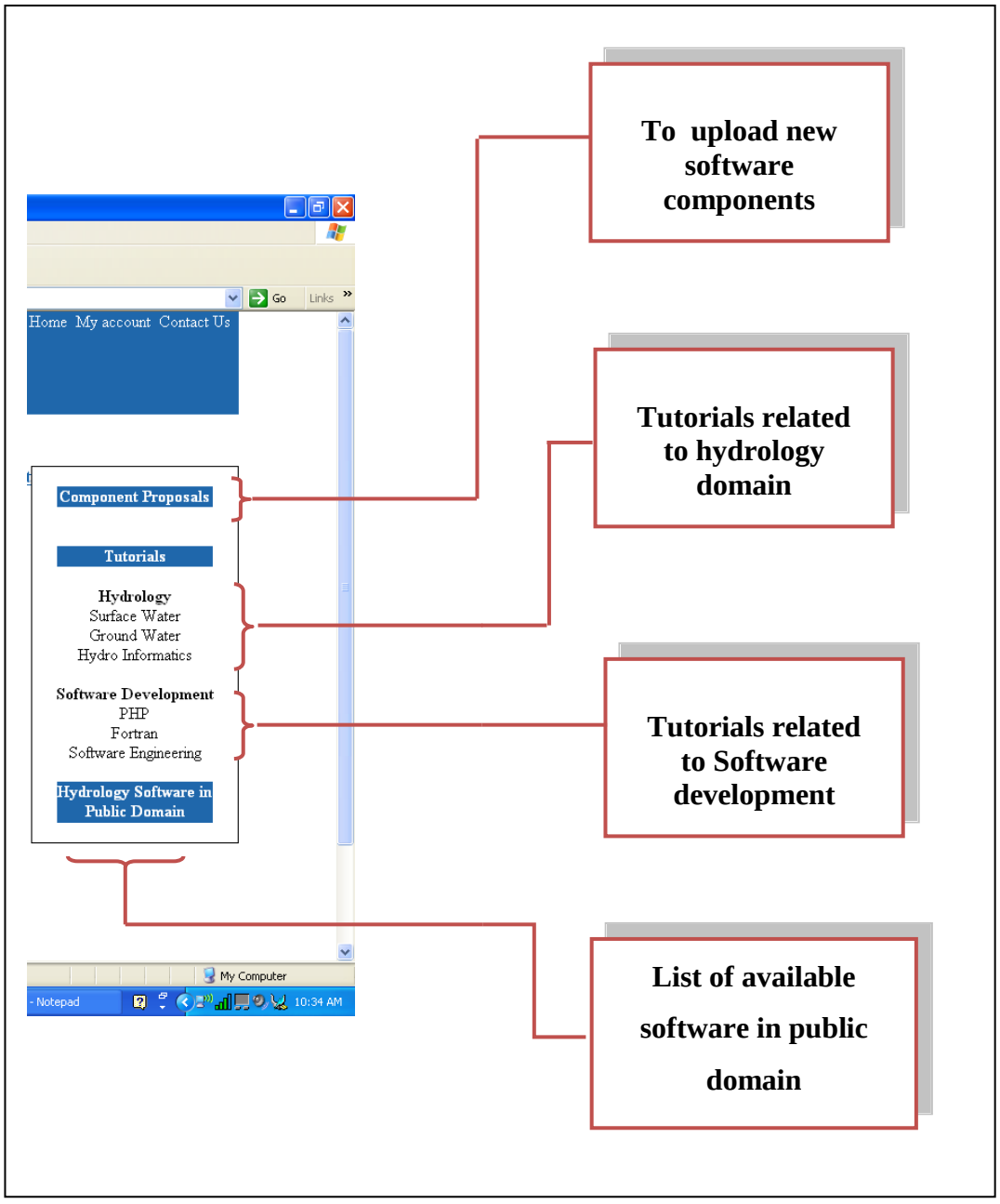


Figure 69: Supporting feature in the repository

- In addition to ordinary users, the software component repository has a feature of an admin user who takes care of authenticating, allowing or deleting users, monitoring the bug reports and arranging solutions, monitoring the components proposals & coordinating for its development/modification and subsequent upload of its latest release, monitoring & coordinating for the latest & relevant technology topics to be available under the tutorial section. Besides these activities, the admin user takes care of other maintenance issues related to the proper and up-to-date upkeep of the software component repository.

6.9 COMPONENT BASED SOFTWARE DEVELOPMENT

With the available software components in hydrology domain as discussed during the implementation of software component repository, an attempt is being made as follows to discuss a software application in hydrology domain developed using Component-based Software Engineering (CBSE).

The scope of the research work is more focused on software development methodologies and not on solving problems in hydrology, which can be taken up separately using the outcome of this research work. This software application demonstrates the use/reuse of software components and demonstrates the efficacy of CBSE approach in software application development in hydrology domain.

Findings during the studies have revealed that few software frameworks have been developed for hydrological modeling and applications and to solve hydrology domain problems. Some of them are:

1. Object Modeling System(OMS) (Leavesley and Markstrom, 2006)
2. RRMT (Wagener et al., 2001)
3. PREVAH (Viviroli et al., 2009)

4. HYDROMAD (Andrews and Croke ., 2011)

All of the above frameworks are used to solve complex problems and are based upon model development, but are not using best practices of software engineering. More importantly, none of the above four use Component-based Software Engineering (CBSE) approach.

Object Modeling System (OMS) suggested for modeling purposes has pure Java on NetBeans as its development platform (David et al, 2002). Precipitation Runoff Modeling System (PRMS) originally written in FORTRAN was reintegrated through OMS. Model integration through OMS has been observed to be a little slower than the original FORTRAN implementation. To improve OMS, lots of modifications have been done recently, like introducing lightweight implementation of the Java Collections API of OMS has been released. Still it needs further modifications (Kralisch and Krause, 2005).

HYDROMAD introduced the concepts of top-down modeling of catchment hydrology models and Rainfall Runoff Modeling. It is implemented on open source R system. R is an environment and a language for graphs and computation statistics (R Development CoreTeam, 2010). Primary mode of R System is command based rather then pointing & clicking. It provides little support of software components (Andrews and coke , 2011).

RRMT is Rainfall Runoff Modeling Toolbox develop in MATLAB environment. RRMT is a closed-source product. Users have neither the freedom to adapt the code nor to share it.

Like Object Modeling System, PREVAH, HYDROMAD and RRMT, this research work has also used implemented hydrology's rain fall run off model, using reusable Software components from the PHP based software component repository. Software components selected for the software application so developed are described as follows:

1. **User Interface**: This software component has been designed for providing Web-based user interface. End users can input data on a screen with following parameters/fields: Hydraulic conductivity Ksat (cm/hr), Soil Parameter Porosity, Soil Water Retention, Soil Moisture, Green-Ampt Parameter, etc. This component is designed in such a way, so that it can be used in any software that requires data with similar parameters.

2. **Time Series:** This non visual software component is a middleware designed to import time series data from end users through an interface provided for the purpose. This component then internally transfers this data to rainfall runoff model. Sample of Time series data is given in the following table.

Table 12: Sample Time Series Data

Date	**Time**	**Stage**	**Camp 1 Rain**	**Camp 2 Rain**	**Camp 3 Rain**
dd/mm/yy	hh/mm/ss	(cms)	(mm)	(mm)	(mm)
3-Jul-04	6:30	-27	0	0	Na
3-Jul-04	6:45	-27	0	0	Na
3-Jul-04	7:00	-27	0	0	Na
3-Jul-04	7:15	-27	0	0.26	Na
3-Jul-04	7:30	-27	0	0.26	Na
3-Jul-04	7:45	-27	0	0	Na
3-Jul-04	8:00	-27	0	0	Na
3-Jul-04	8:15	-27	0	0	Na
3-Jul-04	8:30	-27	0	0	Na
3-Jul-04	8:45	-27	0	0.13	Na
3-Jul-04	9:00	-27	0	0.13	Na

3. **Rainfall Runoff modeling:** This component provides the implementation of the functionality of runoff model. The implementation involves mathematical calculations based on amount of rainfall in its catchment, amount of water in the watersheds, and amount of water in drainage basins.

   It basically calculates the water in rainfall, watersheds and drainage into runoff. The component provides an interface to capture the input, performs aforementioned calculations and produces an output which then becomes an input for the Hydrograph component, described as a next component. The flowchart (David, 2003) as below figure 70 depicts the steps followed by this component to calculate infiltration and runoff.

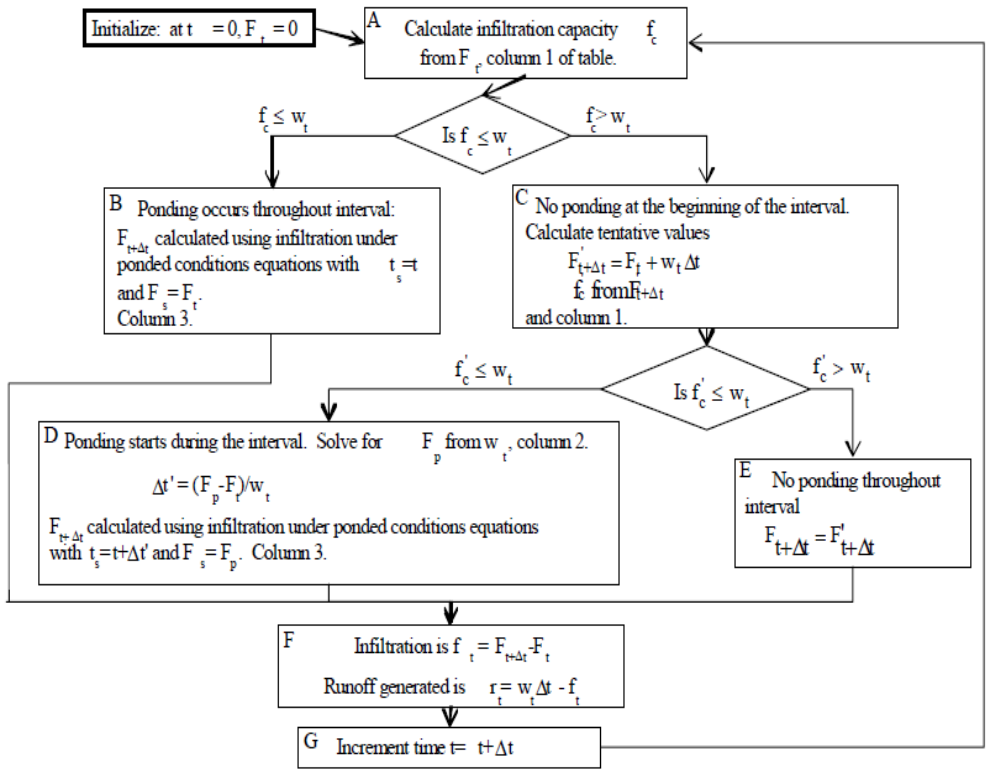


Figure 70: Flow chart to calculate infiltration and runoff

Table 13 :Equations for Rainfall Runoff model

	Infiltration Capacity	Cumulative infiltration at ponding	Cumulative infiltration under ponded condition
Green-Ampt Parameters and			Solve implicitly for .

4. **Hydrograph:** This software component draws a hydrograph on the client's web browser based on the inputs provided in the form of infiltration and runoff data calculated during the rainfall-runoff model's calculations performed in the component described earlier.

Software application Rain fall runoff process is developed through Component-based software engineering approach with the four reusable components. User interface is describe in figure 71.

In this web based user interface following input are taken

[1] Hydraulic conductivity K_{sat} (cm/hr),

[2] Soli Parameter Porosity,

[3] Soil Water Retention,

[4] Soil Moisture

[5] Green-Ampt Parameter and

[6] Data file that contain time series data.

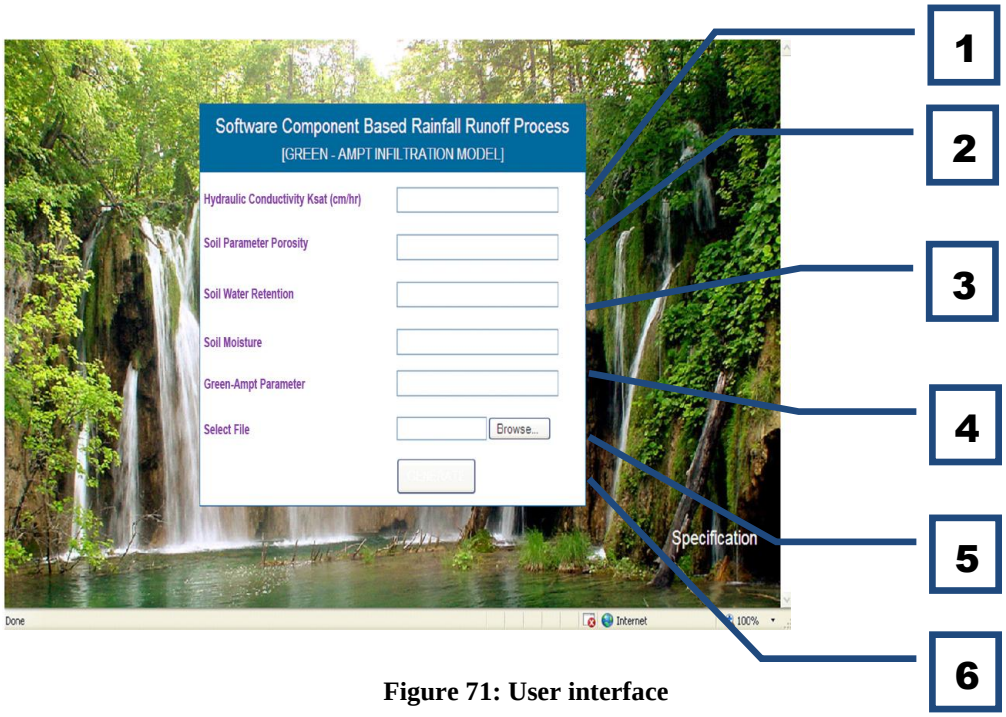


Figure 71: User interface

To import time series data to model, the file name is specified through file chose dialog box. File contain time series data of rainfall and it will be passed as text file to model, as shown in next figure 72.

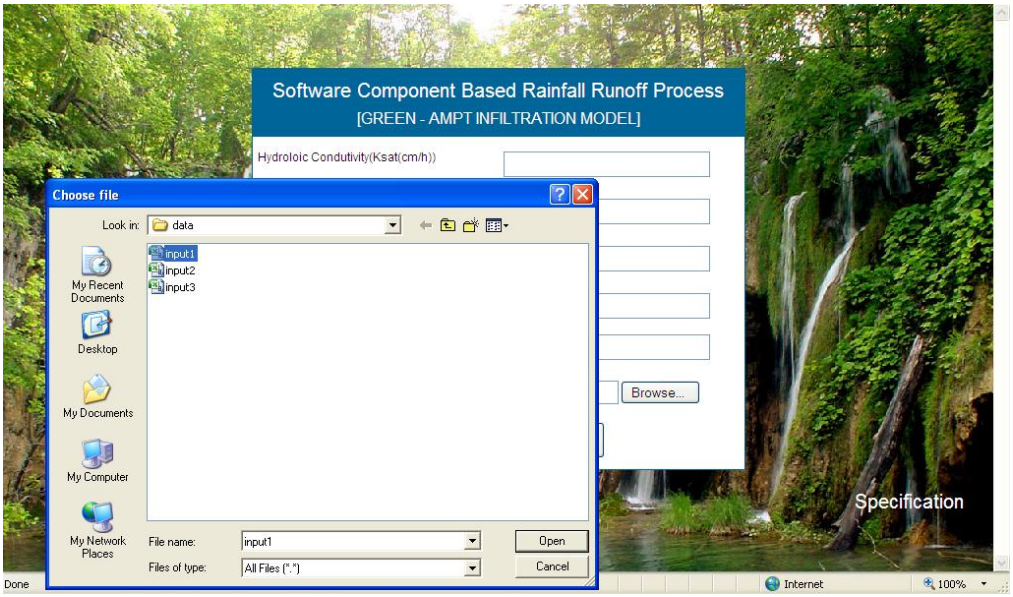


Figure 72: Time Series data import to model

Input data will pass into rainfall runoff model , which calculate runoff and infiltration. After that output went to hydrograph software component which hydrograph against different data sets shown in next figures 73.

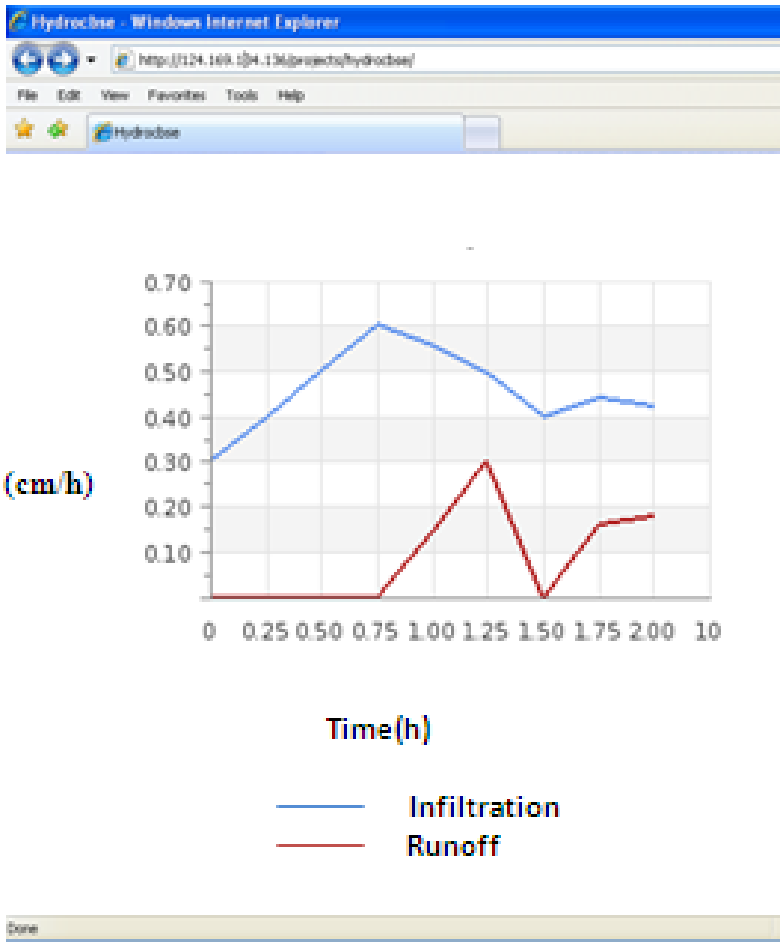


Figure 73: Hydrograph Corresponding data Set II

Figure 74 shows the hydrograph that represent Infiltration and Runoff corresponding Rainfall data set II.

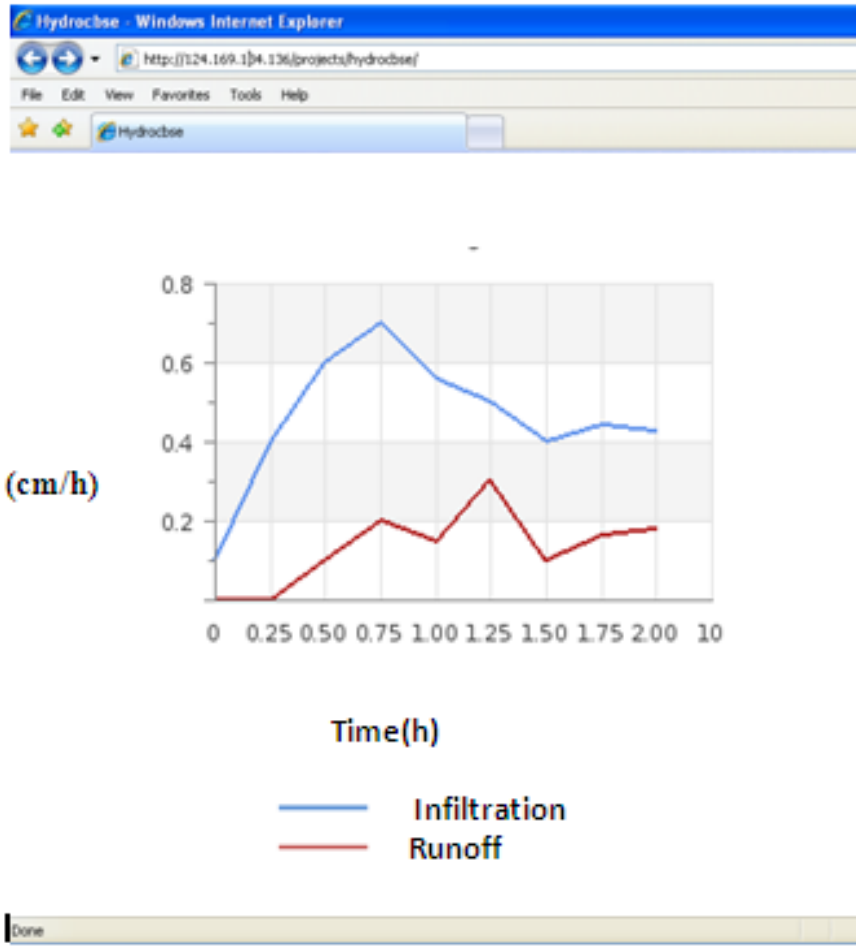


Figure 74 : Hydrograph Corresponding data Set II

6.10 SUMMARY

This chapter discussed feasibility of reusable software component in surface water area of hydrology domain. After that, a domain specific Component-based software engineering framework was proposed for development of software components and application development through software component. Web-based software component repository also developed to demonstrate use of CBSE

approach in domain specific software applications. Following are some of the salient features the software component repository:

- It is designed as a web-based one, so that it can be accessible 24x7 - anywhere, anytime through Internet.
- Following open source software has been used:
    - **IDE:** Eclipse
    - **Development language:** PHP
    - **Database:** MySql
- Authenticated and controlled access has been provided to authorized users to the software component repository, in which new users can register themselves & get an authorized used-id & password. A mechanism has also been provided for the users to retrieve their passwords, if forgotten.
- The software components developed and uploaded in the repository are meant for
    - Generation of hydrograph,
    - Making mathematical calculations & logical analysis of two types of rainfall runoff model, viz., Tank model and Green ampt infiltration model,
    - Providing data import utility,
    - Generating an input form, and
    - web service to present weather report.

- Download option, which when clicked provides the facility to download available software components on the local client machine use/reuse any of the software components for developing software applications.

- Documentation option also provided, which when clicked provides full documentation support containing details, like specifications etc. for each of the software components.

- Bug Update option, which clicked provides the facility to report bugs in any of the software component observed during its implementation.

- The right-hand column has an option to submit proposals of new/modified components for the repository.

- Tutorials on two areas – one on hydrology related topics, like Surface Water, Ground Water, Hydro Informatics, and the other tutorial is related to software development technologies, like PHP, FORTRAN and Software Engineering, either of which when clicked provides detailed learning to the user on that particular technology.

- Link to public domain software in hydrology are provided.

- Finally this software repository is used for design & development of rain fall run off process through G reen Ampt Infiltration model.

Next chapter concludes the thesis and presents the recommendations.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

"I think and think for months and years .Ninety-nine times, the conclusion is false. The hundredth time I am right"
Albert Einstein (1879-1955)

This chapter briefly recapitulates salient findings of this research work, presents the recommendations and gives pointers to future research in this area.

Component-based Software Engineering (CBSE) is an upcoming area in Software Engineering. Its applications in a particular domain are not yet understood clearly. To study the use of Component-based Software Engineering approach in a domain, extensive literature was reviewed, particularly in the areas of Component-based software development issues and trends, software reusability, software applications in hydrology domain. I summarize the research objectives sought to be addressed in this research work within the overall context of Component based Software Engineering in domain prospective. These objectives were:

> **Objective 1:** To analyze and compare Software Components Models being used in Component Based Software Engineering (CBSE).

Objective 2: To discover the role of Component-Based Software Engineering in the software being used in a domain (Hydrology domain as an example).

Objective 3: To design a framework for software components development and Component-Based Software Engineering approach for domain specific applications.

7.1 REVIEW OF EFFORTS AND RESULTS OF THE RESEARCH

Objective 1: To analyze and compare Software Components Models used in Component Based Software Engineering (CBSE).

Existing work has been discussed before classifying and analyzing the software component models. Earlier classifications have been done on software components architecture description languages-ADL (Medvidovic and Taylor 2000), software component model (Lau and Wang 2007), added functional properties (Crnkovic et al 2005), in particular business area (Kotonya et al. 2003). Knowledge based classification of models was also proposed by Kotonya where research categorized different models on several aspects like process, product, role, business domains, COTS and technology (Kotonya et al. 2003).

On business solution software components another classification has been done on following set of properties like reuse, interoperability, visibility, granularity, state and use of Object Oriented approach (Fellner and Turowski 2000).

13 software component models are surveyed and analyzed by Lau and Wang (2007) who provided a taxonomy based desiderata for CBSE based software development.

For my work on classification and analysis, 32 software component models were analyzed. Section 3.6 of this thesis includes the list of these 32 software

component models. It incorporated a combination of general purposes as well as domain specific models. Software component models in different domains have been classified and analyzed on the following parameters:

1. **Interface Specification and Language Support**

2. **Nature of software component model - Domain Specific or General**

On the basis of exercise of classification and analysis of software components, following findings emerged:

Interface Specification

Software components models have two types of interface specifications:- operation-based and port-based. Operation-based interfaces have methods with parameters as a interface signature and port-based interfaces have input and output port for communications. Most of software component models for embedded system have port based interfaces. Few software components models provide both types of interfaces. Interface specifications are mandatory part of software component models. Table 14 exhibits interface based classification.

Table 14: Interface Based Classification

Interface Specification	Software Component Models
Port Based	BlueArX, COMDES II, PIN, PECOS, SaveCCM, PRBUS, ROBOCOP, ProCOM,
Operation based	EJB, Fractal, Koala, KobrA, Java Bean, MSCOM, OpenCOM, OSGi, Pallado, Sofa
Both	CompNETS, CCM, BIP, AUTOSAR.

Language Support

It was observed that a number of languages are being used in software component models as shown below:

Table 15: Language Support

Language	Percentage
Domain Specific Language	12
C Language	23
Java Language	17
CORBA IDL	8
Microsoft IDL	
UML	
XML	
Coco, Prolog	4
PYTHON BASE	
Robocop IDL	
Pascal	

C language is being used in the maximum number of software component models.

Nature of software component model - Domain Specific or General

Next classification was based upon the nature of software component models Domain Specific or General. To developed domain specific applications, system requirements vary from domain to domain. One Software component model may not be fit for other domain applications. 32 software component models were used for this analysis and classification.

There are a limited number of component models which are exclusive to a domain, except for embedded systems. For example, domains like Earth Modeling, Robotics, Automobiles Industries, and HPC have software component

models designed and developed exclusively for the needs of these particular domains only.

Fig 75 exhibits the distribution of software component models across domains.

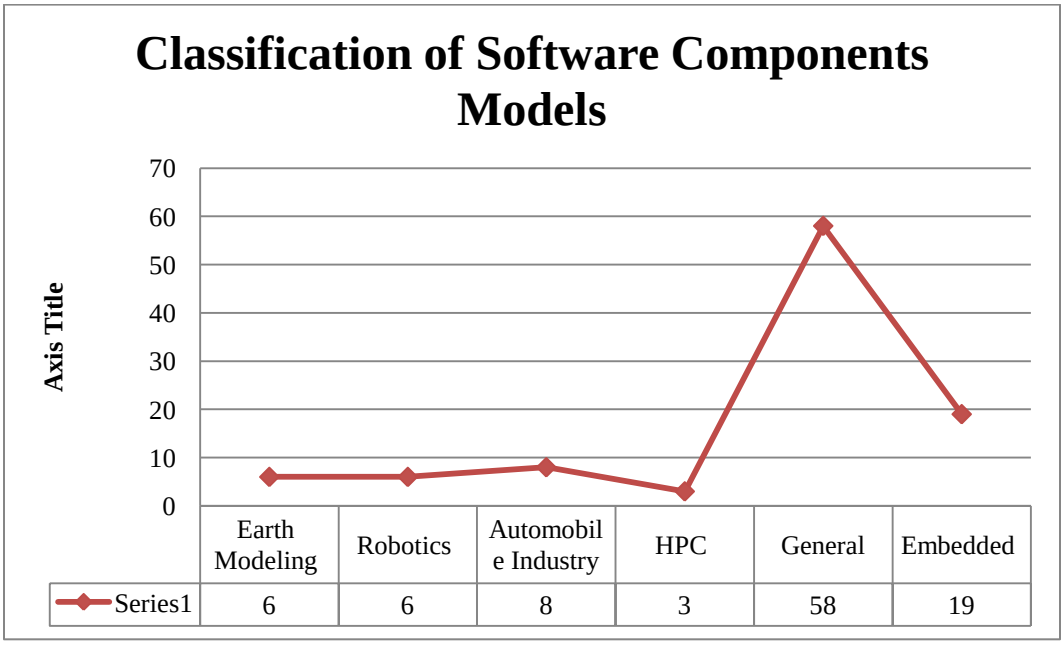


Figure 75 : Classification of Software Components

Most of the software component models are being used for general purposes and are not exclusive to a domain. These general purposes models can be configured, re-configured and adapted for use among many domains.

The domain of embedded systems has a large number of components designed and developed exclusively for the needs of a large number of sub domains like electronics, mobile communications and even in automobiles.

Objective 2: To discover the role of Component-Based Software Engineering in the software used in a domain. To achieve this objective first the use of IT in hydrology domain was reviewed. Main areas where IT plays a part in hydrology are as follows:

Systems concept: The hydrology cycle can be symbolized as a system and a sub system. This will be the base for studying hydrology from IT prospective. Hydrology system cycle has three subsystems: the atmospheric water sub system, the surface water sub system, and the subsurface water sub system. Flow of water is there from one subsystem to other.

Information Exchange: IT plays very important role in flow of information in hydrology system. In most of the cases flow of information in this domain is as follows: Data is coming from Earth Observation, Data logger then it is passed to different models like Physical Modeling, Data Modeling, Agent based Model to Knowledge management that deal mostly with integration with hydrologic and hydraulic models to Decision Support.

Data Integration: Hydrology related data is collected from different locations by many state and central Government organizations and is shared by many agencies. Data in scientific computing in hydrology domain is heterogeneous in nature and is collected from different sources. Now-a-days, IT plays a very important role in data integration. For instance, Indian Meteorology department (IMD) uses the data captured by weather satellites for weather forecasting integrating it in their own forecasting models. Its output is being further utilized by other agencies in their own systems.

Decision Support System: Needs Assessment Report to Government of India's for Hydrology Project II has suggested usage of Decision Support System (DSS) for hydrology. The report has outlined many successful cases in support of the applicability of DSS as Integrated Water Resources Development and Management System that would fulfill the requirement of all the states and centre organizations dealing with hydrology. It is one of the main area where IT is extensively used for example data management, model integration, scenario management, analysis management and various reports analysis.

e-Learning: Domain experts from hydrology are starting to use e-learning systems as a better way of exchange of ideas. Many learning packages have been developed by national as well as international organizations. These packages cover important elements including various definitions and terminologies used in hydrology. The idea behind the development of such packages is to provide a flexible learning environment and a platform to exchange knowledge in water domain. Considering the importance of e-Learning, National Institute of Hydrology (NIH) India has developed software named '*Learning Package for Hydrology'*.

Modeling Paradigms are extensively used in hydrology and will continue to be used in future also. There are lots of commercially sophisticated modeling packages available today developed by many organizations. There are three main type of modeling paradigms in hydrology like process based modeling, data-driven modeling and agent based modeling. Details of use of IT in hydrology domain including systems concept, information exchange, data integration, decision support system, e-learning and modeling paradigms etc. are given in section 5.1 and 5.2.

With regard to objective 2, major efforts and findings are as follows:

Use of software in the domain with respect to Indian context was explored and a survey was carried out to find out the usage status of software, development methodologies, source of software used, areas where software is being used, use of CUAHSI Web Services etc. The findings from the survey are as follows:-

- It was found that software in hydrology domain in India is being used by State, Central Government Organizations, and Academics & Research institutes.
- Separate lists were compiled showing the various software developed by third parties that are in use in the hydrology domain. A listing was also

done for in-house developed software during Hydrology Project II of GOI and other in-house developed software by Indian organizations.

- Most of the use of software is in sub areas of hydrology like surface water and ground water. Domain experts also used some software in both the areas as exhibited in Figure 76.

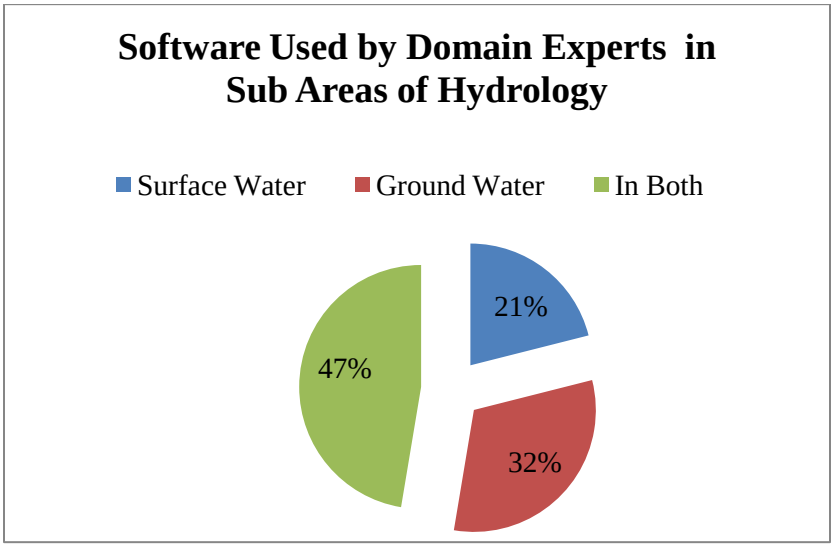


Figure 76: Software Usage Areas

- Another finding of the survey was that there was very little awareness about the latest and best software development practices in the in software application developed for hydrology domain. For example, software reusability through OOA, CBSE and Software Product Lines, etc It was not being used barring a few cases in some of the research institutes (15%-20%), but not through technology like software components.

Objective 3: To design a framework for software components development and Component-Based Software Engineering approach for domain specific applications.

One of the application areas of the domain, surface water was identified for further study in this research work. 40 software were studied to find out the use of software reusability using CBSE by using an experimental setup, each with a computer machine and operating system confirming to the requirements of the software-to-be-analyzed. List of software used in experiment are in Annexure 2. The finding of the experimental setup was that there was a very limited application of software reusability through Component-based software engineering in hydrology's surface water domain-specific software under consideration. Partial use of non-component based reusable code was however found in some applications like OpenMI, RRL Library. However these were not taken into consideration as they were not using the Component-based Software Engineering paradigm.

Two important application areas were selected out of five possible areas for this research work to identify the possibility of reusable software components through components based software engineering approach. "Data integration and management" was selected as it was the only one common across all other application areas. The second area the "Rainfall Runoff process" was chosen as an example application area due to its relevance to the Indian context. Exhibits of possible reusable software components are in section 6.2.

To understand how can software components be used/ reused in domain, this research work explored and has proposed a domain specific Component-based Software Engineering framework that includes the development of software components as well as development of application from these software components. This framework is available in the public domain. However, due to

the transient nature of the Web, it is suggested that the researcher be contacted for the latest location of the application.

A web-based software components repository has been developed as a framework. This framework provides a skeleton structure in which one can imbibe software components both domain specific and general purpose for the use of water scientists, engineers, hydrologists, researchers and water managers to refer to, use and reuse them whenever there is a need of designing, developing, augmenting or modifying domain-specific software applications.

The underlying platform for this framework uses the LAMP stack (Linux, Apache, MySQL & PHP) running on an Eclipse instance. By using the framework one can easily develop domain specific application through reusable software component repository based upon proposed framework. User can also upload their software components in the repository to give access to other developer.

The details of usage of this framework are available in Section 6.8. As sample set of eight components has already been developed and incorporated into this framework.

To demonstrate the applicability of CBSE to the Hydrology domain, an application has been designed and developed to implement the Rainfall Runoff model using four reusable software components available in the framework repository. This includes the user interface, data import, rainfall run off model and hydrograph. This application generates a hydrograph to represent infiltration and runoff. It has been tested extensively on different datasets of rain fall sourced from rain gauge stations for Hebehalla watershed in Bandipur National Park India.

The above mentioned software components can be reused in other applications also as per the need there by corroborating the validity of the developed framework.

To recapitulate, the findings of the research work are summarized in the table below:

Table 16: Summary of Major findings

Objective	Findings
Objective 1: To analyze and compare Software Components Models being used in Component Based Software Engineering (CBSE).	1. 32 software component models found being used in industry, research and academia. 2. These software components model are analyzed on Interface Specification and Language Support, Nature of software component model - Domain Specific or Normal. 3. Java (19%) and c (23%) is dominating other language by providing support to most of software component model even domain specific language is position 3 with 12%. 4. Limited domain specific software component models exist. 5. Software component models being used in domain like Earth Modeling, Robotics, Automobiles Industries, and HPC domain etc
Objective 2: To discover the role of Component-Based Software Engineering in the software being used in domain	1. In Indian context Software used by State, Central Government Organizations, and Academics & Research institutes. 2. Software usage has been classified into two

(Hydrology domain as an example).	categories - in-house developed and commercially available application software.. 3. Exhibits for software used in different areas of hydrology. 4. 40 software for hydrology surface water domain were evaluated. Software reusability through OOA, CBSE or Software Product Lines was not found in any of these software. 5. Partial use of reusability concept will be existing small areas and it's not through software components.
Objective 3: To design a framework for software components development and Component-Based Software Engineering approach for domain specific applications	1. Reusable proposed software components are identified from two application areas of hydrology domain 2. Proposed framework for development of software components in hydrology. 3. Design and Developed web based software component repository. 4. Developed application through these reusable software components.

7.2 RECOMMENDATIONS

This section presents the recommendations being made based on the findings of this research as follows:

Although the Component-based software engineering one of the promising area of Software engineering and large or small scale application are developed through it to reduced development cost, time, complexity, defect density and increased the quality software product, software reusability etc. In domain specific applications, role of CBSE is limited that restricts to take various advantages offered by it.

So it is recommended that CBSE approach be used in Hydrology domain as well that would result in quality software applications being developed for the water scientists, engineers, hydrologists, researchers and water managers.

Further, use of CBSE approach in the hydrology domain would also help in improving the software development process and reducing the time and cost required for development of software applications in this domain.

The web-based software component repository is a ready reference for developing and deploying domain-specific software applications 'just-in-time', or 'on-the-fly' especially by domain experts having limited software development experience in the areas of Rain fall runoff and Data integration. It is recommended that the use of software component repository be extended for other application areas in hydrology such as:

- River Flow & Flood Forecasting.
- Models and Work Flow Management.
- Statistical Analysis.

7.3 LIMITATIONS OF THIS RESEARCH

This research tried to understand use of Component-based Software Engineering approach in the identified domain. Following were the limitations encountered by the researcher in completing this work.

- Well documented and published technical details are available only for few software components technologies. It is one of the main challenges for this work.
- Development methodologies are not disclosed by the concerned software companies due to their closed source nature, proprietary and other legal issues.
- Software Engineering is not the strength of hydrology domain experts, as a result of which there is very little awareness about the latest developments and best practices in the area of software application development, for example software reusability through OOA, CBSE and Software Product Lines, etc.
- There is limited number of domain-specific software developed by Indian software companies.

7.4 SUGGESTIONS OF FUTURE WORK

This research studied the use of CBSE in a domain, future researchers can take this research as a basis to carry out research in following areas.

- The Domain specific CBSE framework developed in this research work may be implemented in the related organizations, not only for their own needs but also for mutual professional cooperation & sharing. Software component repository will be promoted, augmented and further

strengthened through frequent usage and by addition of new domain specific components.

- The developed Domain Specific CBSE framework may be ported to other domains.

- The web-based software component repository may be extended further and deployed through a public or a private cloud so as to take advantages of cloud computing using SaaS or PaaS models as exist today.

- Further development in software components should be in the direction to design and developed Component as a Service (CaaS) another service on cloud platform to contribute in development of software application.

- CaaS will be another service on cloud computing like, Platform as a Service (PaaS), Infrastructure as a Service (IaaS) etc. This will enable CaaS to boost the economy of the flailing software component Industry.

REFERENCES

Abebe, A. J., Guinot, V. and Solomatine, D. P. (2000). "Fuzzy alpha-cut vs. Monte Carlo techniques in assessing uncertainty in model parameters", Proc. 4th International Conference on Hydroinformatics, Iowa City, USA, July 2000.

Abrahart, R. J. and See, L. (2000). "Comparing neural network and autoregressive moving average techniques for the provision of continuous river flow forecast in two contrasting catchments.", Hydrological processes 14: 2157-2172.

Abrahart, R. J. and See, L. M. (2007). "Neural network modelling of non-linear hydrological relationships. Hydrol.", Earth System. Sci. 11.: 1563–1579.

Advant(2011) OCS Open Control System
www05.abb.com/.../3bse004616r0201_l_en_advant_ocs_-_s100_i_o_system.pdf

Alonso, G., F. Casati, H., Kuno and Machiraju, V. (2004). Web Services: Concepts, Architectures and Applications, Springer-Verlag.

Andrews, F. T., B. F. W. Croke and A. J. Jakeman (2011). "An open software environment for hydrological model assessment and development.", Environ. Model. Softw. 26(10): 1171-1185.

Androutsellis-Theotokis, S., Spinellis, D., Kechagia, M. and Gousios, G. (2011). "Open Source Software: A Survey from 10,000 Feet", Foundations and Trends® in Technology, Information and OM 4(3-4): 187–347.

Armstrong,R.(2005)."High-Performance Scientific Component Research: Accomplishments and Future Directions."
www.cca-forum.org/db/news/documentation/whitepaper05.pdf

Armstrong, R. and Kumfert, G. (2006). "The CCA component model for high-performance scientific computing.", Concurrency and Computation: Practice and Experience 18(2): 215-229.

Atkinson, C., Bayer, J., Bunse, C., Kamsties, E., Laitenberger, O., Laqua, R., Muthig, D., Paech, B., and Zettel, J. (2001). Component-Based Product Line Engineering with UML, Addison-Wesley.

Auer Ken and Miller Roy, Extreme Programming Applied: Playing to Win, Addison-Wesley Professional-2001

Bachmann, F., Bas, L., Buhman, C., Comella-Dorda, S., Long, F., Robert, J., Seacord, R. and Wallnau K. (2000). "Technical Concepts of Component-Based Software Engineering.", Technical Report CMU/ SEI -2000 –TR -008, Software Eng. Inst., Carnegie Mellon University 2.

Baiju, M. (2009). "A Comprehensive Guide to Zope Component Architecture.", 2011, from http://www.lulu.com/content/1561045.

Basili, V. R. and Boehm, B. W. (2001). "COTS-Based Systems Top 10 List.", IEEE Computer 34(5): 91-93.

Basili, V. R., Rombach, D., Schneider, K., Kitchenham, B., Pfahl D. and R. W. S. (Eds.) (2008). "Empirical Software Engineering Issues.", Proc. Int'l. Workshop Dagstuhl, Germany, Springer . LNCS 4336.

Basili, V. R., Rombach, H. D., Basili, V. R. and Selby, R. (1993). The experimental paradigm in software engineering. In Experimental Software Engineering Issues: Critical Assessment and Future Directives, Proc of Dagstuhl - Workshop, published as Lecture Notes in Computer Science #706, Springer-Verlag.

Bass, L., Ivers, J., Klein, M. and Merson, P. (2005). "Reasoning Frameworks.", (CMU/SEI-2005-TR-007). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005.

Bastide, R. and Barboni, E. (2004). "Component-Based Behavioural Modelling with High-Level Petri Nets.", Proc. Third Workshop Modelling of Objects, Components and Agents.

Basu, A., Bozg, M. and Sifakis, J. (2006). "Modeling Heterogeneous Real Time Components in Bip.", Proc. Fourth IEEE Int'l Conf. Software Eng. and Formal Methods.

Becker, S., Koziolek, H. and Reussner, R. (2007). "Model-Based Performance Prediction with the Palladio Component Model.", Proc. Sixth Int'l Workshop Software and Performance, 2007 ,54-65.

Bentley.(2011). http://www.bentley.com/

Bentley and MJP (2011). WaterGEMS® Powers Design of Break through 24-by-7 Water System in India. Technical Case Study, http://www.bentley.com/fr-FR/Corporate/Publications/Case+Studies/ Water + and +Wastewater/MJP.htm

Bentley and NMMC (2011). "Navi Mumbai Deploys Bentley Software to Meet Demand for Public Services.", Technical Case, http://www.bentley.com/en-US/Solutions/Water+and+Wastewater/Case+ Studies.htm

Beran, B. and Goodall, J. (2009). Standardizing Access to Hydrologic Data Repositories through Web Services, IEEE International Conference on Advanced Geographic Information Systems & Web Services.2009

Bhattacharya, B. and Solomatine, D. P. (2000). Application of artificial neural network in stage-discharge relationship, Proc. 4th International Conference on Hydroinformatics, Iowa City, USA, July 2000.

Bigot, J., Bouziane, H., Pérez, C., Priol, T., César, E., Alexander, M., Streit, A., Träff, J., Cérin, C., Knüpfer, A., Kranzlmüller D. and Jha, S. (2009). On Abstractions of Software Component Models for Scientific Applications, Euro-Par 2008 Workshops - Parallel Processing, Springer Berlin / Heidelberg. 5415: 438-449.

Binbin, Q., Qian, L. and Yansheng, L. (2010). A framework for dynamic analysis dependency in component-based system, 2nd International Conference on Computer Engineering and Technology (ICCET), 2010.

Body, R. (2011). An off the shelf flood forecasting system, In the Proceedings of 2011 Computing and Control for the Water Industry (CCWI) Conference, Exeter, United Kingdom. .

Booch, G. (1996). Object Oriented Analysis and Design with Applications, 2nd ed. Benjamin Cummings Publishing Company Inc.ISBN 0-8053-5340-2.

Box. D (1998). Essential COM, Addison-Wesley,1998.

Brian, F.(2012) "Software Crisis 2.0.", IEEE Computer Society Press Los Alamitos, CA, USA 45(4): 89-91.

Brooks, A., Kaupp, T., Makarenko, A., Williams, S. and Oreback, A. (2005). Towards component-based robotics, In: Proc. IROS-05, IEEE Press.

Broy M, Henn J, Koskimies. K , Plasil, F., Pomberger, G., Pree, W., Stal, M. and Szyperski, C. (1998). "What Characterizes a Software Component?", Software Concepts and Tools vol. 19(no. 1): pp. 49-56.

Bruneton, E., T. Coupaye, M. Leclerc, V. Quéma and J.-B. Stefani (2006). "The Fractal Component Model and its Support in Java.", Software Practice and Experience. 36(11-12): 1257-1284.

Bruneton, E., Coupaye, T. and Leclercq, M. (2004). An Open Component Model and Its Support in Java, Proc. Seventh Int'l Symp. Component-Based Software Eng. (CBSE '04).

Bruneton, E., Coupaye, T. and Stefani, J. (2003). "The Fractal Component Model.", Object Web Consortium, Technical Report Specification 2.

Cheesman, J. and Daniels, J. (2000). UML Components: A Simple Process for Specifying Component-Based Software, Addison-Wesley. 2000

Cheng-Hua and Leu (1998). The joint application of remote sensing and GIS technologies to network-based hydrologic modeling, IEEE International Geosciences and Remote Sensing Symposium Proceedings, IGARSS 1998.

Clarke, M., Blair, G. , Coulson, G. and Parlavantzas N. (2001). An Efficient Component Model for the Construction of Adaptive Middleware, Proc. IFIP/ACM Int'l Conf. Distributed Systems Platforms.

Collins, N., Theurich, G., DeLuca, C., Suarez, M., Trayanov, A., Balaji, V., Li, P. , Yang, W., Hill, C. and Silva, A. d. (2005). " Design and Implementation of Components in the Earth System Modeling Framework.", International Journal of High Performance Computing Applications 19(3): 341-350.

ComponentFactory(2012), "Component Factory" http://www.componentfactory.com/

ComponentSource (2012), " Why to buy from Component Source", http://www.componentsource.com/index.html

Conradi, R., O. P. N. S. J. Li, Bunse C., Torchiano, M. and Morisio, M. (2005). Reflections on conducting an international CBSE survey in ICT industry, Proc. of the 4th International Symposium on Empirical Software Engineering (ISESE) November 17-18, 2005, Noosa Heads, Australia, IEEE Press, pp. 214-223.

Contreras, C., Dubus, J., Dumoulin, C., Flissi, A., Merle, P., Moroy, J. and Rouvoy, R. (2012). "OpenCCM User's Guide.", 2012, from http://openccm.objectweb.org/doc/0.8.1/user_guide.html

Creswell, J. W. (2003). Research Design, Qualitative, Quantitative and Mixed Method Approaches, London, UK, Sage Publications

Crnkovic, I. (2003). "Component-based software engineering new challenges in software development", 25th International Conference on Information Technology Interfaces, 2003: 9- 18.

Crnkovic, I. (2004). "Component-based approach for embedded systems." 9th International Workshop on Component-Oriented Programming 2004.

Crnkovic, I. and Larsson, M. (2002a). "Building Reliable Component-Based Software Systems.", Artech House Inc.

Crnkovic, I. and Larsson, M. (2002b). "Challenges of component based development.", Syst. Software 61(3): 201–212.

Crnkovic, I., Larsson, M. and Preiss, O. (2005). "Concerning Predictability in Dependable Component-Based Systems: Classification of Quality Attributes.", Architecting Dependable Systems III: 257-278.

Crnkovic, I., Sentilles, S., Vulgarakis, A. and Chaudron, M. R. V. (2011). "A Classification Framework for Software Component Models.", IEEE Transactions on Software Engineering 37(5): 593-615.

Crnkovic, I., Stig, L. and Michel, C. (2005). "Component-based Development Process and Component Lifecycle.", Journal of Computing and Information Technology - CIT 4 321-327.

Cuahsi(2012), “CUAHSI Mission”, http://www.cuahsi.org/mission.html

Czarnecki, K. and Eisenecker, U. W. (2000). "Generative Programming: Methods, Tools, and Applications ", Addison Wesley.

Darcy, D. P., Kemerer, C. F., Slaughter, S. A. and Tomayko, J. E. (2005). "The structural complexity of software an experimental test", IEEE Transactions on Software Engineering 31(11): 982 - 995

David G. Tarboton (2003), Rainfall-Runoff Process, Utah State University, 2003

David, O., Markstrom, S. L., Rojas, K. W., Ahuja L. R. and Schneider, I. W. (2002). "The Object Modeling System, in: Agricultural System Models in Field Research and Technology Transfer.", Lewis Publishers, CRC Press LLC.: 317–331.

Davison, R., Martinsons, M. G. and Kock, N. (2004). "Principles of canonical action research.", Information Systems Journal 14(1): 65-86.

Deltares (2012). "http://www.deltares.nl/en/about-deltares."

Dibike, Y. B., Solomatine, D. P., Velickov, S. and Abbott, M. B. (2001). " Model induction with support vector machines: Introduction and applications.", ASCE J of Computing in Civil Engineering, July 2001, 15(3): 208-216.

Emadi, S. and Shams, F. (2008). From UML component diagram to an executable model based on Petri Nets, IEEE International Symposium on Information Technology, 2008. ITSim 2008.

Emmerich, W. and Kaveh, N. (2002). Component technologies: Java beans, COM, CORBA, RMI, EJB and the CORBA component model , Proceedings of the 24th IEEE International Conference on Software Engineering, ICSE 2002.

Endres, A. and Rombach, D. (2003). A Handbook of Software and System Engineering, Empirical Observations, Laws and Theories, Addison-Wesley Professional 2003.

Englander, R. (1997). Developing Java Beans, O'Reilly Media.

Falconer, R., Lin, B. and Harpin, R. (2005). "Environmental modelling in river basin management.", J. of River Basin Management 3(3): 169-184.

Favaro.J, Favaro,K and Favaro, P (1998). "Value Based Software Reuse Investment.", Annals of Software Eng 5: 5-52.

Fellner, K. J. and Turowski, K. (2000). "Classification Framework for Business Components.", Proc. 33rd Hawaii Int'l Conf. System Sciences, 88047.

Frakes, William B. and Favaro, John (2011). Software reuse and safety, Proceedings of the 12th international conference on Top productivity through software reuse. Pohang, South Korea, Springer-Verlag 2011.

Frakesand.W and Terry.C (1996). "Software Reuse: Metrics and Models.", ACM Computing Surveys 28: .415-435

FSI. (1985-2012). "Free Software Foundation.", Retrieved Feb 2012, from www.fsf.org.

Fürst, S. (2010). AUTOSAR Technical Overview, 2nd AUTOSAR Open Conference, Tokyo, Japan.

Gao, j. (2000). "Monitoring Software Components and Component-Based Software.", The 24th Annual International Computer Software & Applications: 403-412.

Genssler, T., Christoph, A., Schulz, B., Winter, M., Stich, C. M., Müller, C. Z. P., Stelter, A., Nierstrasz, O., Ducasse, S., Arevalo, G., Roel, R. Wuyts, P. Liang, B. Schönhage and Born, R. V. D. (2002). "PECOS in a Nutshell" , Last Access 2011.

Gieryan, T. F. (1999). Cultural Boundaries of Science: Credibility on the line, University of Chicago Press.

Glass, R. L. (1998). "Is There Really a Software Crisis?.", IEEE Software 15(1): 104–105.

Glass, R. L., Ramesh, V. and Iris, V. (2004). "An Analysis of Research in Computing Disciplines.", Communications of the ACM 47(6): 89-94.

GNU. (2005). 2011, from www.gnu.org.

Group, C. C. A. F. T. W. (2010). A Hands On Guide to the Common Component Architecture, www.cca-forum.org/download/tutorial/guide-0.7.1-0.pdf, Last Access Jan 2012

Goswami, Puneet, Bhatia, P. K, and Hooda V. (2009). "Effort estimation in Component-based software Engineering.", International Journal of Information Technology and Knowledge Management 2 (2): 437-440.

Hammerschmidt and Christoph (2007). AutoSar standard not ready to plug-and-play, EE Times Europe.

Hanninen, K., Maki-Turja, J., Nolin, M., Lindberg, M., Lundback, J. and Lundback, K. (2008). "The Rubus Component Model for Resource Constrained Real-Time Systems.", Proc. Int'l Symp. Industrial Embedded Systems: 177-183.

Hansson, H., Åkerholm, M., Crnkovic, I. and Törngren, M. (nd). "SaveCCM – a component model for safety-critical real-time systems."

Happe, J., H. Koziolek and R. Reussner (2011). "Facilitating performance predictions using software components." IEEE Computer.

Harmel, R. D., Smith, D. R., King, K. W. and Slade, R. M. (2009). "Estimating storm discharge and water quality data uncertainty: A software tool for monitoring and modeling applications.", Environmental Modeling & amp; Software 24(7): 832-842.

Hauge, Ø., Sørensen, C.-F. and Conradi¸ R. (2009). Adoption of Open Source in The Software Industry, Proc. IFIP WG 2.13 Conference on Open Source - Development, Communities and Quality (OSS 2008), Milan, Italia, Springer Verlag.

Heinecke, H. (2004). AUTomotive Open System ARchitecture An Industry-Wide Initiative to Manage the Complexity of Emerging Automotive E/E-Architectures, In Proceedings of the Convergence International Congress & Exposition On Transportation Electronics, Detroit, MI, USA.

Heineman, G. and Councill, W. (2001). Component-Based Software Engineering: Putting the Pieces Together, Addison-Wesley, 2001.

Hidalga, A. N. d. l., Zhao, L. and Sampaio, P. R. F. (2008). "Domain Engineering Approach for the Development of Tendering E-Marketplace Applications ", IEEE International Conference on Computer Science and Software Engineering: 927-933.

Hill, C., DeLuca, C., Balaji, V., Suarez, M. and Silva, A. d. (2004). "Architecture of the Earth System Modeling Framework.", Computing in Science and Engineering. 6(1): 18-28.

Hissam, S., Ivers, J., Plakosh, D. and Wallnau, K. C. (2005). Pin Component Technology (V1.0) and Its C Interface, Software Engineering Institute, Carnegie Mellon University.

Ho-Cheng, L., Yi-Haur, S. , Chen-Pey, H., Jyh-Horng, W. and Whey-Fone, T. (2004). The integration and application of the computational grid structure in flood forecast system, Proceedings. Seventh International Conference on High Performance Computing and Grid in Asia Pacific Region, 2004..

Hoffer, J. A., George, J. F., Valacich, J. S. and Upper Saddle, N. (2005). Modern Systems Analysis, Prentice Hall.

Hong, S. and F. J. Lerch (2002). "A Laboratory Study of Consumers' Preferences and Purchasing Behavior with Regards to Software Components." The DATA BASE for Advances in Information Systems 33.3: 23-37.

Hopkins, J. (2000). "Component primer.", Communications of the ACM 43:10: 27-30.

Horsburgh, J. S., Tarboton, D. G., Schreuders, K. A. T., Maidment, D. R., Zaslavsky, I. and Valentine, D. (2010). Hydroserver: A Platform for Publishing Space Time Hydrologic Datasets, AWRA Spring Specialty Conference Geographic Information Systems (GIS) and Water Resources, Orlando Florida, America.

HP.(2011)."Hydrology Project II :About Us.", from http://www.hydrology-project.gov.in/AboutUs.htm.

Hsu, K. L., Gupta, H. V. and Sorooshian, S. (1995). "Artificial neural network modeling of the rainfall-runoff process.", Wat. Res. 31(10): 2517-2530.

Hu, T., Wu, F. and Zhang, X. (2007). "Rainfall–runoff modeling using principal component analysis and neural network.", Nordic Hydrology 38(3): 235-248.

Huang, L., Guo, B., Jiang, C. and Wei, Y. (2009). The Research and Design for Web Print Component Based on EJB Technology, Second International IEEE Conference on Information and Computing Science, 2009. ICIC '09.

Huang, X. and Ouyang, H. (2010). "EDI of Hydrological Data Collection Platform.", IEEE International Conference on Computer Application and System Modeling (ICCASM 2010) VI: 186-190.

Hutchinson, J. and Kotonya, G. (2006). A Review of Negotiation Techniques in Component Based Software Engineering, 32nd EUROMICRO Conference on Software Engineering and Advanced Applications, 2006. SEAA '06. .

IEEE (1990). "IEEE Standard Glossary of Software Engineering Terminology.", IEEE Std 610.12-1990: 67.

Iribarne, L. (nd). "Web Components: A Comparison between Web Services and Software Components.", Columbian Journal of Computation, June 2004

Jain, H., Vitharana, P. and Zahedi, F. M. (2003). "An Assessment Model for Requirements Identification in Component-Based Software Development" The DATA BASE for Advances in Information Systems 34(4): 48-63.

Jalote, P. (May 2008). A Concise Introduction of Software Engineering, Springer - Verlag London Limited 2008.

Jamal, H. (2011). "Engineering Hydrology.", 2012, from http://www.aboutcivil.com/hydrology.html.

Jian, S., Andrew, P. and John, R. (2005). "A new approach for a Windows-based watershed modeling system based on a database-supporting architecture.", Environ. Model. Software. 20(9): 1127-1138.

Jinhyun, K., Jin-Young, C., Inhye, K. and Insup, L. (2011). "Generating composite behavior of embedded software components based on UML behavioral model and process algebra.", SIGSOFT Software. Eng. Notes 36(1): 1-9.

Jonkers, L. (2001). "Personal communication."

Junzhong, J., Jingyue, L., Reidar, C., Chunnian, L., Jianqiang, M. and Weibing, C. (2008). Some lessons learned in conducting software engineering surveys in china, Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement. Kaiserslautern, Germany, ACM.

Kakola, T. (2012). Introduction to Software Product Lines: Engineering, Service, and Management Minitrack, 45th Hawaii International Conference on System Science (HICSS 2012).

Kang, K., Cohen,S., Hess,J., Novak,W., Peterson: (November 1990). "Feature-oriented domain analysis (FODA) feasibility study.", Technical Report CMU/SEI-90-TR-021 , Software Engineering institute, Carnegie- Mellon University.

Kang, W. F. a. K. (2005). ""Software Reuse Research: Status and Future", IEEE Transactions on Software Engineering vol. 31, no. 7: pp 529-536.

Ke, X., Sierszecki, K. and Angelov, C. (2007). COMDES-II: A Component Based Framework for Generative Development of Distributed Real-Time Control Systems, Proc. 13th IEEE Int'l Conf. Embedded and Real-Time Computing Systems and Applications.

Kerholm, M., Carlson, J., Fredriksson, J., Hansson, H., Ha˚kans-son, A. Mo¨ller, P. Pettersson and Tivoli, M. (2007). "The SAVE Approach to Component-Based Development of Vehicular Systems.", J. Systems and Software, 80(5): 655-667.

Khu, S.-T., Savic D., Liu, Y., and Madsen, H. (2004) . "A fast evolutionary-based meta modeling approach for the calibration of a rainfall-runoff model.", 2nd Biennial Meeting of the International Environmental Modelling and Software Society, iEMSs: Manno, Switzerland.

Ki, K. C. and Eisenecker, U. W. (2000). Generative Programming: Methods Tools and Applications., Addison-Wesley, 2000.

Kim, I., Bae, D. and Hong, J. (2007). "A Component Composition Model Providing Dynamic, Flexible, and Hierarchical Composition of Components for Supporting Software Evolution", The Journal of Systems and Software 80(11): 1797.

Kim, J. E., Rogalla, O., Kramer, S. and Haman, A. (2009)., Extracting, Specifying and Predicting Software System Properties in Component Based Real-Time Embedded Software Development, Proc. 31st Int'l Conf. Software Eng., 2009.

Kotonya, G., Sommerville, I. and Hall, S. (2003). Towards a Classification Model for Component-Based Software Engineering Research, Proc. 29th EUROMICRO Conf.: 43-52.

Kralisch, S., Krause P., and David, O. (2005). "Using the object modeling system for hydrological model development and application.", Advances in Geosciences 4: 75-81.

Kukko, M., Helander, N. and Virtanen, P. (2008). Knowledge Management in Renewing Software Development Processes, Hawaii, Proceedings of the 41st Annual International Conference on System Sciences,.

Kum, D. K. and Kim, S. D. (2006). A Systematic Method to Generate .NET Components from MDA/PSM for Pervasive Service, Fourth IEEE International Conference on Software Engineering Research, Management and Applications, 2006.

Kung-Kiu, L. (2006). Software component models, Proceedings of the 28th international conference on Software engineering. Shanghai, China, ACM.

Kung-Kiu, L., Ling, L. and Perla Velasco, E. (2007). Towards composing software components in both design and deployment phases, Proceedings of the 10th international conference on Component-based software engineering. Medford, MA, USA, Springer-Verlag.

Kung-Kiu, L. and Zheng, W. (2007). "Software Component Models.", Software Engineering, IEEE Transactions on 33(10): 709-724.

Kung-Kiu Lau, M. a. Z. W. (2006). "A Software Component Model and its Preliminary Formalization" Springer-Verlag Berlin Heidelberg.

Kwon, T. and Z. Su (2010). "Automatic Detection of Unsafe Dynamic Component Loadings." IEEE Transactions on Software Engineering PP(99): 1-1.

Larsson, I. C. a. M. (2000). "A Case Study: Demands on Component-based Development." ACM ICSE 2000: pg 23-31.

Lau, K.-K., Ornaghi, M. and Wang, Z. (2005). "A software component model and its preliminary formalization." , In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.-P. (eds.) FMCO 2005. Springer, Heidelberg (2006) 4111: 1–21.

Lau, K.-K., F. Taweel, G. Lewis, I. Poernomo and C. Hofmeister (2009). Domain-Specific Software Component Models, Component-Based Software Engineering, Springer Berlin / Heidelberg. 5582: 19-35.

Lau, K.-K. and Wang, Z. (2007). "Software Component Models." IEEE Trans. Software Eng 33(10): 709-724.

Leavesley, G. H., Markstrom, S. L. and Viger, R. J. (2006). "USGS Modular Modeling System (MMS) AS Precipitation - Runoff Modeling System (PRMS)." ,CRC Press: 159-177.

Li, J., Conradi, R., Mohagheghi, P. , Sæhle, O. A., Wang, Ø., Naalsund, E., and Walseth, O. A. (2004). An Empirical Study on Component Reuse inside IT industries, Proc. 5th Int'l Conf. on Product Focused Software Process Improvement (PROFES'2004) Kyoto, Japan, Springer Verlag, LNCS.

Lin, J.-W., Lai, Y.-C., Peng, H.-H. and Wu, W. (2006). A Web-based Environmental Experiment System Using Microsoft .NET and COM , Proceedings of the IEEE Instrumentation and Measurement Technology Conference, 2006. IMTC 2006.

Lobbrecht, A. and Solomatine, D. P. (1999). "Control of water levels in polder areas using neural networks and fuzzy adaptive systems." , Water Industry Systems: Modelling and optimization applications: eds D Savic and G Walters, Research Studies Press Ltd 2: 509-518.

Luqi and Zhang, L. (2007). "Software Component Repositories.", Wiley Encyclopedia of Computer Science and Engineering. .

Maaskant, H. (2004). "A Robust Component Model for Consumer Electronic Products." , Philips Research Book Series 3: 167-192.

Maaskant, H. (2005). "A Robust Component Model for Consumer Electronic Products." , Springer 3: 167-192.

Madanmohan, T. R. and De, R. (2004). "Open Source Reuse in Commercial Firms." , IEEE Software 21(1): 62-69.

Martin, L. G. (1994). Software reuse experience at Hewlett-Packard, Proceedings of the 16th International conference on Software engineering. Sorrento, Italy, IEEE Computer Society Press.

Medvidovic, N. and Taylor, R. N. (2000). "A Classification and Comparison Framework for Software Architecture Description Languages.", IEEE Trans. Software Eng 26(1): 70-93.

Mehta, A. (2001). "Evolving Legacy Systems Using Feature Engineering and CBSE." , IEEE Computer: 797-798.

Min Yang and Jung (2011). A layered approach for identifying systematic faults of component-based software systems, Proceedings of the 16th international workshop on Component-oriented programming. Boulder, Colorado, USA, ACM.

Minns, A. W. and Hall, M. J. (1996). "Artificial Neural Network as Rainfall-Runoff Model.", Hydrological science Journal 41(3): 399-417.

Moore, R. (2010a). "OpenMI Document Series: OpenMI Standard 2.0 Interface, Specification for the OpenMI (Version 2.0).", OpenMI Document Series.

Moore, R. (2010b). "The OpenMI in Nutshell (version 2)." The OpenMI Document Series.

Natan, R. (1995). CORBA: A Guide to Common Object Request Broker Architecture, McGraw-Hill.

Newcomer, E. (2002). Understanding Web Services: XML, WSDL, SOAP, and UDDI., Addison-Wesley.

Niemueller, T., Ferrein, A., Eckel, G., Pirro, D., Podbregar, P., Kellner, T., Rath C., and Steinbauer, G. (2010). "Providing Ground-truth Data for the Nao Robot Platform.", Robo Cup Symposium 2010.

Oberndorf, T. (1997). "COTS and Open Systems - An Overview." Retrieved May, 2011, from http://www.sei.cmu.edu/str/descriptions/cots.html#ndi.

OMG (2003). "Unified Modelling Language Specification.", Object Management Group .

OMG (2011) "Common Object Request Broker Architecture (CORBA) Specification 3.2"., Release date November 2011 www.omg.org/spec/CORBA/3.2/

Ommering, R. v., Linden, F. v. d., Kramer, J. and Magee, J. (2000). "The Koala Component Model for Consumer Electronics Software.", IEEE Computer: 78-86.

OMS. (2011). "Object Modeling System (OMS).", from http://www.javaforge.com/project/oms.

Oracle. (2012). "Java Bean Specifications ", from http://www.oracle.com/technetwork/java/javase/documentation/spec-136004.html.

OSI. (1998-2012). "Open Source Initiative.", Retrieved Feb. 2012, from http://www.opensource.org/index.php.

Panunzio, M. and Vardanega, T. (2010). A Component Model for On-board Software Applications, 36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA).

Pavelsky, T. M. and Smith, L. C. (2008). "RivWidth: A Software Tool for the Calculation of River Widths From Remotely Sensed Imagery.", Geoscience and Remote Sensing Letters, IEEE 5(1): 70-73.

Payne and Christian (2002). "On the Security of Open Source Software.", Info Systems Journal 12(1): 61-78.

Perrauda, J.-M., Podgerb, G. M., Rahmana, J. M. and Vertessya, R. A. (2003). "A New Rainfall Runoff Software Library.", MODSIM 03.

Peter, H. (1999). SIMS Oliver, Business Component Factory, Wiley.

Philippe, M. (2008). "Fractal: A Component Model for Adaptive Software Systems."

Ping, A., Zhi-jian, W., Xiao-feng, Z. and Yuan-sheng, L. (2003). Architecture of hydrological telemetering software based on Web services, International Conference on Computer Networks and Mobile Computing, 2003. ICCNMC 2003.

Plasil, F., Balek, D. and Janecek, R. (1998). SOFA/DCUP: Architecture for Component Trading and Dynamic Updating, Proc. Fourth Int'l Conf. Configurable Distributed Systems (ICCDS '98).

Qureshi, M. R. J. and Hussain, S. A. (2008). "A reusable software component-based development process model." Advances in Engineering Software 39(2): 88-94.

R Development CoreTeam (2010). "R: A Language and Environment for Statistical Computing.", R Foundation for Statistical Computing, Vienna,Austria.

Ramesh, V., Glass, R. L. and Vessey (2004). "Research in Computer Science: An Empirical Study", Journal of Systems and Software 70(1-2): 165-176.

Ravindran, N. and Yao, L. (2007). HIDE - A Web-based Hydrological Integrated Data Environment, Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, SNPD 2007.

Rebecca and Grinter, E. (2001). From local to global coordination: lessons from software reuse., Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work. Boulder, Colorado, USA, ACM.

Robson, C. (2002). Real World Research: A Resource for Social Scientists and Practitioner-researchers Blackwell Publishers.

Ruffin, M. and Ebert, C. (2004). "Using Open Source Software in Product Development : A Primer.", IEEE Software 21(1): 82-86.

Ruiz, J. L., Duenas, J. C. and Cuadrado, F. (2008). A Service Component Deployment Architecture for e-Banking., 22nd International Conference on Advanced Information Networking and Applications - Workshops, 2008. AINAW 2008.

Schmid., K. (2000). "Scoping Software Product lines. In Patrick Donohoe.", Software Product Lines, Experience and Research Directions, 513–532.

Schmid., K. (2000). "Scoping software product lines. In Patrick Donohoe, Software Product Lines, Experience and Research Directions.", Kluwer Academic Publisher,: 513–532.

Schmidt, D. G. (2000). "An overview of the Real-Time CORBA specification.", IEEE Computer 33 (6): 56-63.

Sedigh-Ali, S., Ghafoor, A. and Paul, R. A. (2001). "Software Engineering Metrics for COTS-Based Systems." , IEEE Computer: 44-50.

SEI (2000) "Domain Engineering: A Model-Based Approach.", Software Engineering Institute, Carnegie-Mellon University.

Seinturier, L., Pessemier, N., Duchien, L. and Coupaye, T. (2006). "A Component Model Engineered with Components and Aspects.", 9th Intl. Symp. on Component-Based Software Engineering (CBSE). 4063: 139-153.

Seker, R. and Tanik, M. M. (2004). "An Information Theoretical Framework for ModelingComponent-Based Systems.", IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews 34(4): 475-484.

Sentilles, S., Vulgarakis, A., Bures, T., Carlson, J. and Crnkovic, I. (2008). A Component Model for Control-Intensive Distributed Embedded Systems, Proc. 11th Int'l Symp. Component Based Software Eng. Oct. 2008.

Server, (2012), "HydroServer" , http://hydroserver.codeple.com

Shanzhen, Y. and Xiang, Z. (2010). Intelligent information sharing in digital watershed modeling for planning and management, 18th International Conference on Geoinformatics, 2010

Sharp, J. H. and Ryan, S. D. (2010). "A Theoretical Framework of Component-Based Software Development Phases ", The DATA BASE for Advances in Information Systems 41: 56-75.

Shaw, M. (2003). Writing Good Software Engineering Research Papers, In Proceedings of the 25th International Conference on Software Engineering.

Sivamuni, K. and Rengaramanujam, S. (2008). "A retrospective on software component quality models.", SIGSOFT Software. Eng. Notes 33(6): 1-10.

Sjøberg, D. I. K., Hannay, J. E., Hansen, O., Kampenes, V. B., Karahasanovic, A., Liborg, N.-K. and Rekdal, A. C. (2005). "A Survey of Controlled Experiments in Software Engineering.", IEEE Trans. Software Eng 31(9): 733-753.

Solomatine, D. P. and Dulal, K. N. (2003). "Model tree as an alternative to neural network in rainfall-runoff modeling.", Journal Hydrological Sciences 48(3): 399-411.

Sommerville, L. (2010). Software Engineering, Addison Wesley.

Sparling, M. (2000). "Lessons Learned Through Six Years of Component-Based Development.", Communications of the ACM 43:10(10): 47-53.

Storey and Sugumaran (2003). "A Semantic-Based Approach to Component Retrieval.", The DATA BASE for Advances in Information Systems 34(3): 8-24.

Szyperski, C. (2002). Component Software, Addison-Wesley 2002.

Szyperski, C. (2011). Componenet Software Beyond Object-Oriented Programming, Pearson 2011.

Taherkordi, A., Eliassen, F., Rouvoy, R., Le-Trung, Q., Meersman, R., Tari, Z. and Herrero, P. (2008). ReWiSe: A New Component Model for Lightweight Software Reconfiguration in Wireless Sensor Networks On the Move to Meaningful Internet Systems, OTM 2008 Workshops, Springer Berlin / Heidelberg. 5333: 415-425.

Tarboton, D. G., Maidment, D. , Zaslavsky, I., Ames, D. P., Goodall, J. and Horsburgh, J. S. (2010). ,"Cuahsi Hydrologic Information System." 2010.

Talby David, Hazzan Orit, Dubinsky Yael and Keren Arie(2006). "Agile Software Testing in a Large-Scale Project", IEEE Software, Special issue on Software Testing - Jul/Aug 2006.

Thorkilsen, M. and Dynesen, C. (2001). "An owner's view of hydroinformatics: Its role in realizing the bridge and tunnel connection between Denmark and Sweden.", J. of Hydroinformatics, IWA Publishing 3(2).

Tichy, W. F. (1998). "Should computer scientists experiment more? 16 reasons to avoid experimentation.", IEEE Computer 31(5).

Tichy, W. F., Lukowicz, P., Prechelt, L. and Heinz, E. A.. (1995). "Experimental evaluation in computer science: A quantitative study", Journal of Systems Software 28(1): 9-18.

Turner, C., Fuggetta, A. and Wolf, A. (1997). Toward Feature Engineering of Software Systems, Technical Report CU-CS-830-97, Department of Computer Science, University of Colorado, Boulder, February 1997.

Vescan, A., Grosan, C. and Pop, H. F. (2008). Evolutionary Algorithms for the Component Selection Problem, 19th International Workshop on Database and Expert Systems Application, 2008. DEXA '08.

Vitharana, P. (2003). "Risks and challenges of component based software development.", ACM Communication 46(8): 67–72.

Vitharana P., Zahedi, F. M. and Jain, H. (2003,). "Design, Retrieval, and Assembly in Component-Based Software Development." , Communications of the ACM . 46(11): 97-102.

Viviroli, D., Zappa, M., Gurtz, J. and Weingartner, R. (2009). "An introduction to the hydrological modelling system PREVAH and its pre- and post-processing-tools." , Environmental Modelling and Software 24(10): 1209-1222.

Vliet:, J. C. (2008). Software Engineering: Principles and Practice, Wiley & Sons.

Voas, J. M. (1998a). "The Challenges of Using COTS Software in Component-Based Development." , IEEE Computer 31(6): 44-45.

Voas, J. M. (1998b). "COTS Software the Economical Choice?", IEEE Software 15(2): 16-19.

Voas, J. M (1999). "Maintaining Component-Based Systems.", IEEE Software 15(4): 22-27.

Voas, J. M. and Miller, K. W. (1995). "Software Testability: The New Verification." , IEEE Software 12 (3): 17-28.

Vriend, H. d. (2011). "Delteres R&D Highlights 2010."

Wagener, T., Boyle, D. P., Lees, M. J., Wheater, H. S., Gupta, H. V. and Sorooshian, S. (2001). "A framework for development and application of hydrological models.", Hydrology and Earth System Sciences 5(1): 13-26.

Waguespack, L. and Schiano, W. T. (2004). "Component-Based IS Architecture." , Information Systems Management 21(3): 53-60.

Wang, Z. (2011). A Software Component Model with Sharing., International Conference on Business Computing and Global Informatization (BCGIN).

Ward-Dutton and Containers, N. (2000). "A Sign Components are Growing Up." , Application Development Trends 7, 1 (January 2000): 41-44.

Weerts, A. H., Schellekens, J. and Weiland, F. S. (2010). "Real Time Geospatial Data Handling and Forecasting: Examples From Delft-FEWS Forecasting Platform/System.", IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 3(3): 386-394.

WenYou, F. and Yan, L. (2009). GIS Technology in the Hydrological Industry, International Conference on Computational Intelligence and Software Engineering, 2009. CiSE 2009. .

Wigley, A., Sutton, M., MacLeod, R., Burbidge, R. and Wheelwright, S. (2003). Microsoft .NET Compact Framework (Core Reference), Microsoft Press.

Xinyu, Z., Li, Z. and Cheng, S. (2009). The Research of the Component-Based Software Engineering., Sixth International Conference on Information Technology: New Generations, 2009. ITNG '09.

Yacoub, S., Ammar, H. and Mili, A. (1999). "A Model for Classifying Component Interfaces.", Proc. Second Int'l Workshop on Component-Based Software Eng. in Conjunction with the 21st Int'l Conf. Software Eng: 17-18.

Yang, J. (2003). "Web Service Componentization.", Communications of the ACM 46(10): 35- 40.

Yin, R. K. (2003). Case Study Research, Design and Methods, Sage pub.

Yu, A. G. (2009). Software Crisis, What Software Crisis? , IEEE International Conference on Management and Service Science, 2009. MASS '09.

Yu, Y., Lu, J., Fernandez-Ramil, J. and Yuan, P. (2007). Comparing Web Services with other Software Components, IEEE International Conference on Web Services, 2007. ICWS 2007.

Zelkowitz, M. V. and Wallace, D. (1997). "Experimental validation in software engineering.", Information and Software Technology , 39(11): 735-744.

Zhou, J., Zhao, D. , Ji, Y. and Liu, J. (2010). Examining OSGi from an ideal enterprise software component model, IEEE International Conference on Software Engineering and Service Sciences (ICSESS) 2010.

ANNEXURE A

Software Component technology has become a key element in the development of large and complex software systems. Our research work focuses on use of Component Based Software Engineering (CBSE) approach in development of software applications in hydrology domain.

Q1. Are you using any software in hydrology domain?

a) Yes b) No

Q2. If yes, then What software are being used, please provide the name of the software, its purpose, vendor, website information.

Sr. No.	**Name of the software**	**Purpose**	**Vendor**	**Website**
1.				
2.				
3.				
4.				

Q3. In case of In house development of software in your organization for hydrology domain, do you use software component technology / CBSE.

a) Yes (details about Software Components used / developed)
b) No

Q4. CUAHSI ,The Consortium of Universities for the Advancement of Hydrologic Sciences, Inc. was established in 2001 to provide support to

hydrologic science research and education and now has a membership of more then one hundred universities.

CUAHSI provide list web services for Hydrology Information System(HIS). In Indian context, do we use this web service, if yes, for what purposes?

a). Yes

b). No

Q5. Any Details CUAHSI Services in India

Q6. Are you aware about software reusability.

Thanks for your valuable Inputs

Name :

Organization:

Mobile No.:

Email id:

ANNEXURE B

List of software used to check software resuablity through software components.

1. Interactive Hydrologic Analyses and Data Management (ANNIE)
2. Software for Branch-Network Dynamic Flow Model (BRANCH)
3. Bridge Scour Data Management System (BSDMS)
4. Culvert Analysis Program (CAP)
5. Channel Geometry Analysis Program(CGAP)
6. Distributed Routing Rainfall-Runoff Model (DR3M)
7. Full EQuations Model (FEQ)
8. Software for Finite-element surface-water modeling system for two-dimensional flow in the horizontal plane (FESWMS-2DH)
9. Software for An unsteady, one-dimensional, open-channel flow model(FourPT)
10. Software for Graphical Constituent Loading Analysis System (GCLAS)
11. Software for GENeration and analysis of model simulation SCeNarios (GenScn)
12. Software for Regional hydrologic regression and NETwork analysis using Generalized Least Squares (GLSNet)
13. Software for USGS Precipitation-Runoff Modeling System (PRMS) and Modular Groundwater Flow Model (MODFLOW-2005)
14. Expert System for Calibration of HSPF (HSPexp)
15. Hydrological Simulation Program-Fortran (HSPF)
16. Hydrograph Separation Program (HYSEP)

.

17. A grid-based, distributed-parameter watershed model to estimate net infiltration below the root zone (INFIL)

18. Input and Output for a Watershed Data Management (WDM) file (IOWDM)

19. Software for Kendall-Theil Robust Line (KTRLine)

20. LOAD ESTimator (LOADEST)

21. A Program for Determination of Error in Individual Discharge Measurements (MEASERR)

22. Software for Computation of total sediment discharge by the modified Einstein procedure (MODEIN)

23. Manning's n value calculation program(NCALC)

24. National Flood Frequency program (NFF)

25. National Streamflow Statistics program (NSS)

26. OTEQ software

27. Software for One-dimensional transport with inflow and storage (OTIS)

28. Software for Flood-frequency analysis based on Bulletin (PEAKFQ)

29. Software for Precipitation Runoff Modeling System (PRMS)

30. Slope-Area Computation Program (SAC)

31. Software for Computation of fluvial sediment discharge(SEDDISCH)

32. Software for Particle-size *statistics* of fluvial sediments (SEDSIZE)

33. StreamStats streamflow statistics web application (StreamStats)

34. Surface-Water Statistics (SWSTAT)

35. Software for Weighted-multiple-linear REGression (WREG)

36. A Computer Model for Water-Surface PROfile Computations(WSPRO)

37. OpenMI

38. Rain fall Runoff Library RRL

39.Unit Hydrograph Applications for Flood Estimation Package-UHPACKI

40. Flood Estimation for Large Catchments Using Deterministic Approach Package-- FLPACK

Source of public domain software :

- The Hydrologic Engineering Center, U.S. Army Corps of Engineers
- Hydrology and Remote Sensing Laboratory & Agricultural Research Service, Department of Agriculture United States.
- Austin University of Texas at Austin.
- United States Geological Survey.
- Indian Institute of Technology Kharagpur
- National Institute of Hydrology