ISSN:0975-9646

Feasibility study on component based software architecture for large scale software systems

P. G. Chaitanya #, Dr. K.V.Ramesh *,
Department of CSE, GIT, Gitam University,Visakhapatnam, Andhra Pradesh, INDIA
* Department of CSE, GIT, Gitam University,Visakhapatnam, Andhra Pradesh, INDIA

Abstract-**Component-Based Software Engineering (CBSE) addresses the development of systems as assembly of components, components as reusable entities, maintenance and upgrading of systems by customizing and replacing such components. The main feature of this approach is interoperability. This requires established methodologies to support the entire component and system lifecycle including technological, organizational, marketing, legal, and other aspects.The goal of CBSE is to standardize and formalize all disciplines supporting the activities related to Component - Based Development (CBD). This paper investigates the whole environment of component-based approach and implements it into Personal Information System (PIS) by developing a common, stable, service calculation software component for various systems running under PIS. Large scale software system and due to its 'hybrid architecture, it needs interoperability to perform some critical business applications such as service calculation.This paper discusses mainly based on the feasibility study of the whole environment of component-based approach, as it introduces more benefits in terms of reusability, flexibility and maintainability.**

***Keywords*— Software Engineering, Component based, Component oriented, interoperability, reusability.**

I. INTRODUCTION

The component-based approach is the most recent approach and will probably mature over the years of the millennium, in a rapid merging with Internet technologies and e-commerce. Right now, component-based development (CBD) is in the leading edge phase. Indeed, there are now a number of technologies appropriate for, and people with experience in the application of CBD.

CBSE [1] has emerged as a technology for rapid assembly of flexible software systems. It combines the elements of software architecture, modular software design, software verification, configuration and development. CBSE is an approach to software development that relies on software reuse. It emerged from the failure of object-oriented development to support effective reuse. Interoperability is the main promise of Component-Based Software Engineering. This study especially aims to reveal interoperability feature of this approach.

The history of software engineering begins from traditional approach that is the first methodology in the software world. Then object-oriented approach came into this world and brought many new useful features. And the last one is-like a return to the correct mean of engineering-component-based approach. In industry there is of course a history of this maturity as illustrated in Fig. 1 and described below [2].

In the 70's the first traditional methodologies were defined. By traditional approach, it is meant that software development using a set of mature and stable technologies, which often include mainframe-based technologies, structured analysis and development techniques, and procedural languages such as COBOL and RPG. Applications often, built using this approach, are used on mainframes.

During the 80's the object oriented approach was expanded to a theory that covered most of the aspects of software development, including testing and project management. Object-oriented approach promises a way for implementing real-world problems to abstractions from which software can be developed effectively. It is a sensible strategy to transform the development of a large, complex super-system into the development of a set of less complicated sub-systems.

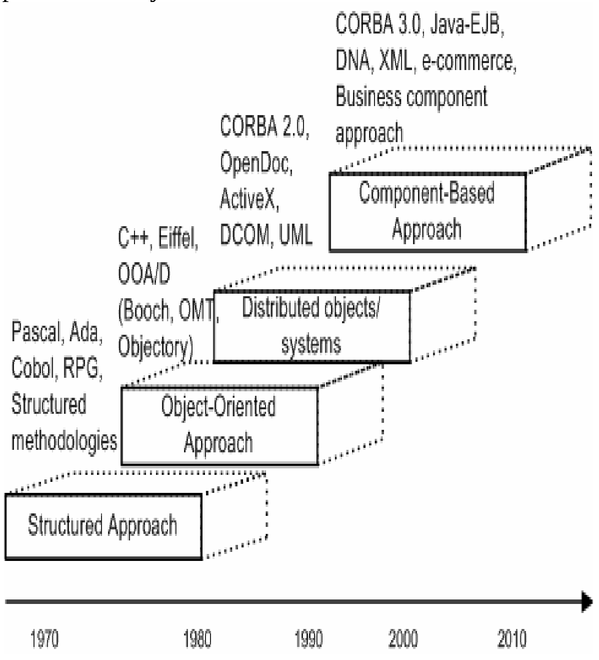
CORBA 3.0, Java-EJB, DNA, XML, e-commerce, Business component approach
CORBA 2.0, OpenDoc, ActiveX, DCOM, UML
C++, Eiffel, OOA/D (Booch, OMT, Objectory)
Pascal, Ada, Cobol, RPG, Structured methodologies
Component-Based Approach
Distributed objects/ systems
Object-Oriented Approach
Structured Approach
1970
1980
1990
2000
2010

Fig .1: The evolution to components in the industry [2].

The main challenge of software today is to manage the complexity and adaptability to the changes. A new approach, focusing on reuse of existing pieces of software and developing reusable entities and based on new component technologies, such as JavaBeans, is becoming dominant. The primary role of Component-based Software Engineering is to deal with developing systems from parts (components), developing parts as reusable entities, and maintaining and improve systems by customizing and replacing those parts. Component-based development (CBD) is, by far, the most promising way of controlling the soaring complexity and cost of business information systems.

II. LITERATURE REVIEW

Iqbaldeep Kaur[1] et.al has done a survey on current component based software technologies and the description of promotion and inhibition factors in CBSE. The features that software components inherit are also discussed. Various attributes are studied and compared keeping in view the study of various existing models for CBSE.

Dr. P. K. Suri[3] et.al emphasized that CBSE approach is actually based on the principle of ‘select and use’ rather than ‘design and test’ as in traditional software methods. He presented a series of papers that cover various important and integral issues in the field concerned.

Arvinder Kaur et.al [4] states that Component based software development approach is based on the idea to develop software systems by selecting appropriate off-the shelf components and then to assemble them with a well-defined software architecture. The key difference between CBSE and traditional software engineering is that, CBSE views the software system as a set of off-the-shelf components integrated within appropriate software architecture. CBSE promotes large-scale reuse, as it focuses on building software systems by assembling off-the-shelf components rather than implementing the entire system from scratch. The author concludes that CBSE is a reuse-based approach to defining and implementing loosely coupled components into systems. The experiences from author’s case studies indicate that our method is feasible in an operational context it improves the efficiency and consistency of evaluations, it has low overhead costs, and it makes the COTS selection decision rationale explicit in the organization.

Arvinder Kaur et.al [5,6] presents an approach for defining evaluation criteria for reusable software components. Author also presents a summary of the common problems in reusable off-the-shelf software selection. Paper discusses that the evaluated aspects of the method are feasible and improve the quality and efficiency of reusable software selection. The off-the-shelf-option (OTSO) method was developed to consolidate some of the best practices that were able to identify for the off-the-shelf (OTS) software selection. The costs involved in adding a component is evaluated by taking into consideration cost involved in selection, identification, evaluation of a component.

Zoran Stojanovic[7] defines a new approach to components through an Integrated Component-Oriented framework that provides a comprehensive component-oriented support for enterprise systems development. The framework should provide flexibility and reduce complexity in the development process and a developed system itself in real application cases where ir should provide efficient and systematic component-oriented system development.

III. COMPONENT - BASED SOFTWARE ENGINEERING

A **component** is a software element that conforms to a software model and can be independently deployed and composed without modification according to a composition standard[3].This approach is expected to revolutionize the development and maintenance of software systems. 70% of new applications will be deployed as a combination of pre-assembled and newly created components integrated to form complex business systems. The main goal of the component - based software engineering is to increase the productivity, quality and interoperability. By using this approach, we can complete programs in less time, language independence, cross-process interoperability, cross-network interoperability.

Component Based Development (CBD) is taken from previous approaches and intermediary approaches such as distributed objects and systems. The distributed object approach extends the object-oriented approach with the ability to call objects across address space boundaries, typically using an “object request broker” capability. The distributed system approach mean a development approach for building systems that are distributed, are often Multi-tier [3].

CBD approach is based on the idea to develop software systems by selecting appropriate off-the-shelf components [3] and then to assemble them with a well-defined software architecture. It is the best way to architect, design, implement, and deploy scalable systems that provide the flexibility and agility required by today’s enterprise environment and is also the latest advance [4] in software development, promising the possibility of extending the real world approach to create well-specified parts and top incorporate legacy code “wrapped” as components.

The purpose of CBD is to develop large systems, incorporating previously developed or existing components, thus cutting down on development time and costs. It can also be used to reduce maintenance associated with the upgrading of large systems. It is assumed that common parts (classes or functions) in a software application only need to be written once and re-used rather than being re-written every time a new application is developed.

CBSE is both a subset and a revolutionary extension of current software engineering practices. Component-based software engineers must define and describe processes to assure timely completion of high quality, complex software systems that are composed of a variety of pre-built software components.

The component-based software life cycle (CSLC) is the life cycle process for a software component with an emphasis on business rules, business process modeling, design, construction, continuous testing, deployment, evolution, and subsequent reuse and maintenance. In general, analysis and design phases for component-based process models take more time than traditional ones take. We expect that the components satisfy flexibility, reusability, contributing to interoperability, and maintainability. Though there are some studies in this era, in this study, only two of them are selected, one of them being from Stojanovic [7] and the other COSE [8].

A. Stojanovic Process Model

A component-oriented development process model, shown in Fig .2, has been introduced by Stojanovic [7], focusing on the component concept from business requirements to implementation. This process will be called by its author's name in this study. The phases of requirements, analysis, design and implementation in a traditional development process has been substituted by service requirements, component identification, component specification, component assembly and deployment.

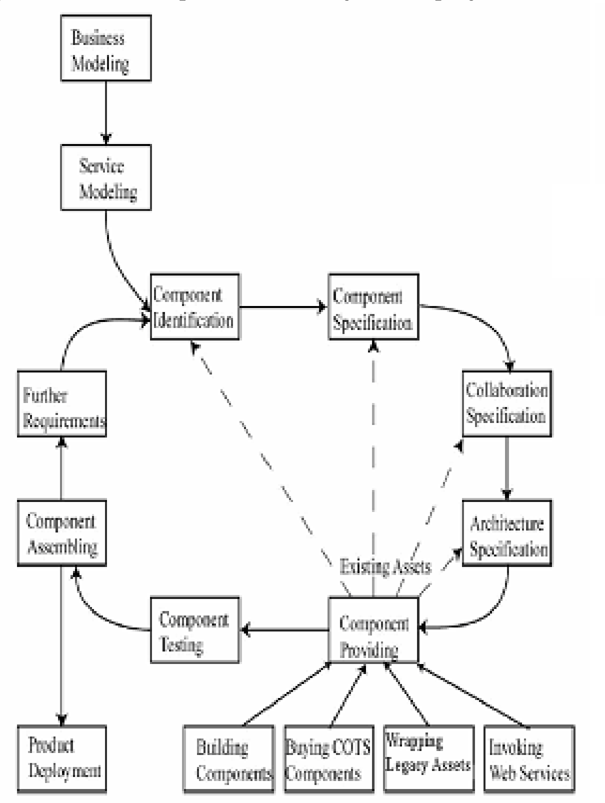

Fig. 2: Stojanovic Process Model

After the components of the system are fully specified, a decision can be made to build components, wrap existing assets, buy COTS (Commercial Off-the-Shelf) components or invoke web services over the Internet.

B. COSE Process Model

Ali H. Dogru et.al [8] emphasized that component-based methodology is immature: Software development methodologies began with traditional approaches that followed the waterfall process model. They then moved toward object-oriented abstractions, which were finally supported by object-oriented methodologies. Now, component-based technology introduces abstraction and lower-level mechanisms but has to be arranged into a comprehensive software engineering process.

The second model, COSE Process Model shown in Fig.3 starts its building activity top-down to introduce the building blocks of the system. As the activity continues towards lower granularity blocks, interfaces between the blocks are also defined. At an arrived level where the module is expected to correspond to a component, a temporary bottom-up approach can be taken; if desired capability can only be achieved by a set of components, their integration into a super-component should be carried out.

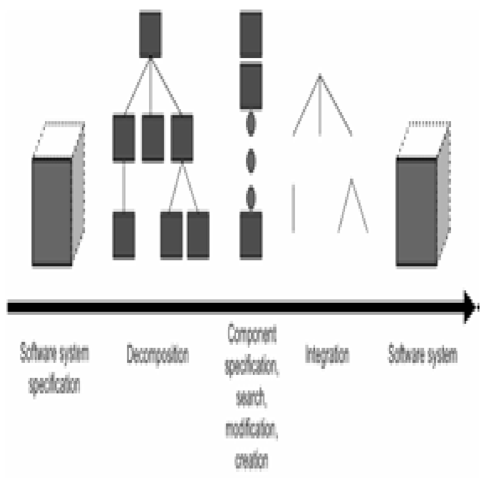
Software system specification
Decomposition
Component specification, search, modification, creation
Integration
Software system

Fig .3: COSE process model

COSE process model consists of four main phases and a system test phase:

• System specification
• System decomposition
• Component Specification, search, modification, creation
• Integration

COSE Process Model starts with *system specification*. Problem is specified and understood, then system high-level requirements are stated and a preliminary search for existing components is conducted in system specification phase. Problem and problem domain knowledge are input to this phase. System high level functional and non-functional requirements and domain related existing component specification documents are the output of this phase.

System is analyzed and decomposed in the system *decomposition* phase. Functional requirements are detailed and required components and their specifications are stated. Components that are going to be implemented in the system are specified and developed in *component specification* phase, either by using existing components or developing new components.

IV. AN ILLUSTRATIVE EXAMPLE

In this paper, a detailed study on various components of Personal Information System (PIS) takes place and focuses on Service Calculation module because of its complexity and methodology in the PIS. For developing Service Calculation module, COSE process model is selected and studied. Each phase of this process model is performed in a step by step manner and these steps are listed below:

A. Specification of Service Calculation

Service Calculation module computes total service period of each personnel starting from beginning date of an employment in the company till today as shown in fig.4. Some specific previous workings, which are again for government, are also added. Previous workings that can be considered under guarantee of any other insurance are also added. Besides, some types of leavings during working are deducted completely. All functionalities, stated above, are considered as independent components. This study proposes that there is a need to develop a common Service Calculation module with the help of component-based methodology.

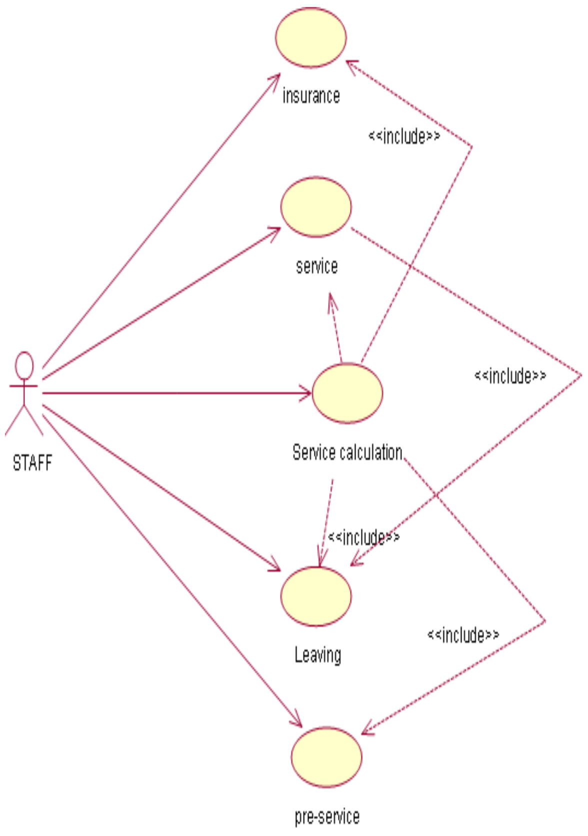

Fig .4: use-case diagram for PIS

B. Decomposition of Service Calculation

This phase is performed primarily using top-down decomposition. Main functions of the system are determined and they correspond to abstractions of high-level system functionalities as shown in fig.5. According to these functionalities, the system is decomposed until a set of terminal package is reached, where none of which can usefully be decomposed Into packages further. As mentioned at specification phase there are six various high-level system functionalities such as normal service period, pre-service periods, insurance periods, and leaving periods. These main functions need various database accesses in different ways and formats so that they need to be constructed separately.

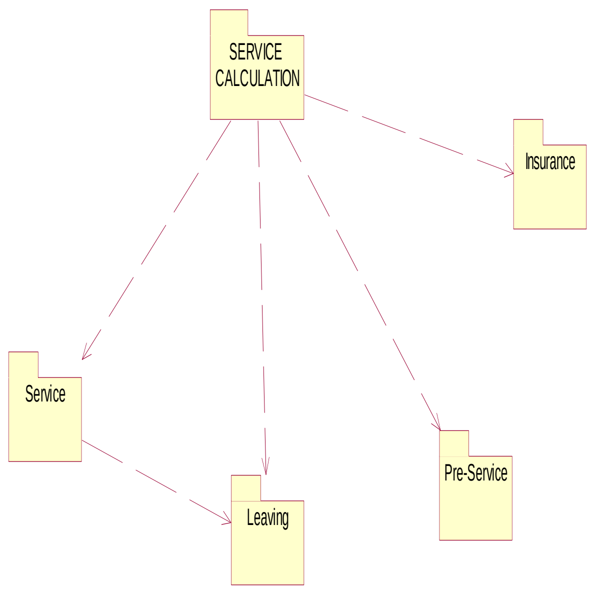

Fig. 5**:** First cut of Service Calculation in COSEML

The components, given are considered the most effective and efficient in terms of reusability and maintainability.

C. Component specification of service calculation

In this phase, logically specified and decomposed system will be transformed into physical entities, that is, components. Searching, modification and creation processes are performed based on the requirements. For this study, searching is an unnecessary process because there has not been any component repository for PBS. Therefore, besides searching, also modification cannot be performed in this study either. In this study, decomposed components listed below are to be developed. The components are Insurance , Service, Pre-Services, Leaving, and Service Calculation

D. Integration for Service Calculation

In this study, our component repository for PIS has only five components. This can be considered as framework. These five components are integrated using their interfaces that work as abstract connection. This integration carries out final service calculation. There is a sequence diagram fig.6 of service calculation component. This phase aims to reach target software system in general. Leaving component is used by only Service component so when service component is activated for specific personnel then Leaving component is activated by Service component's setting function. Other three components are activated for the same personnel and they return relevant Information.

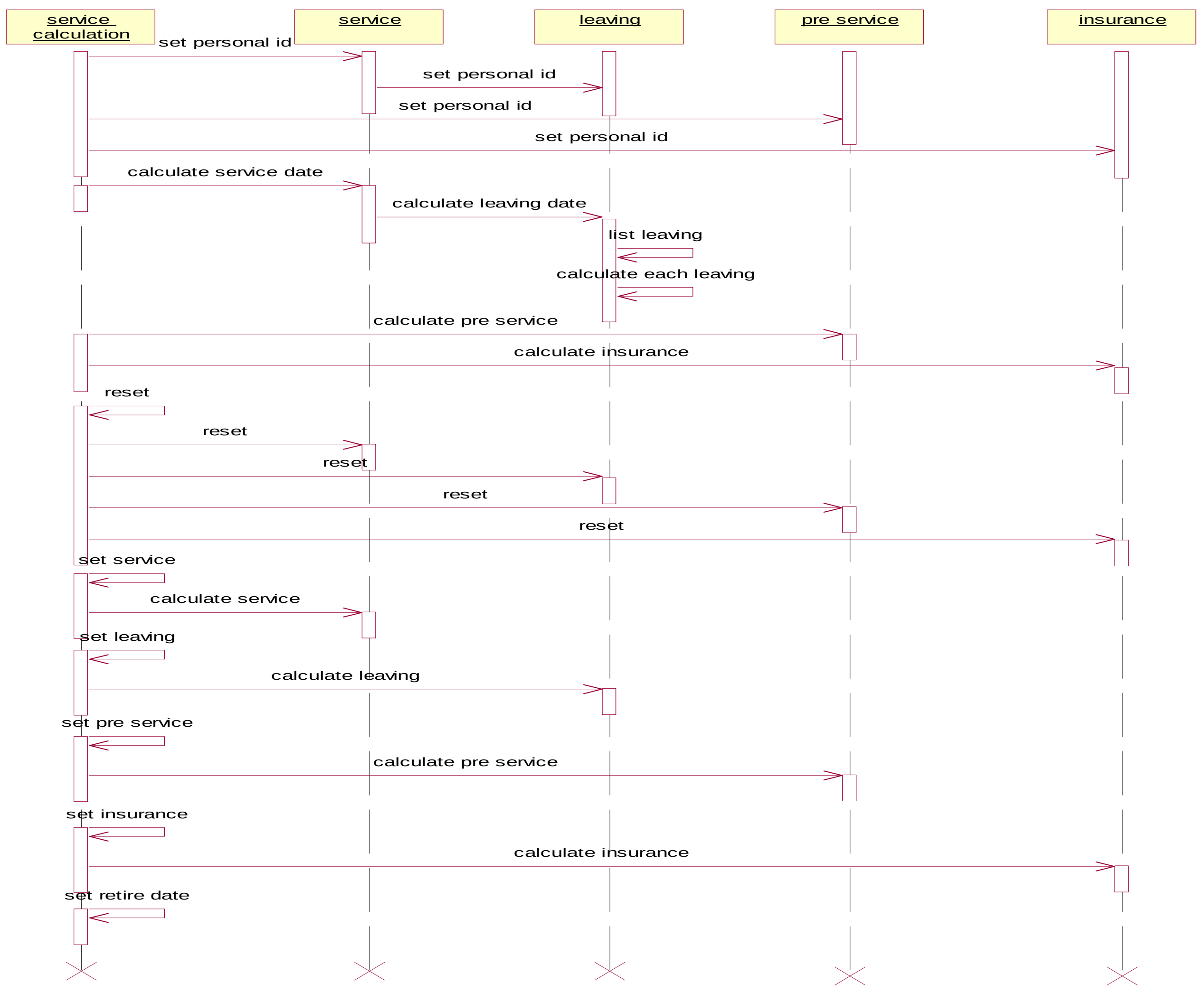

Fig. 6: Sequence diagram for PIS

V. CONCLUSION

This study is implemented a selected module of PIS using a new approach of Software Engineering, namely the component-based Software Engineering. This implementation provides interoperability and reusability between two different systems i.e. PIS and vehicle hire system(VHS) with the access of a common components and avoids duplications for the systems. With this study, maintenance cost is expected to be reduced since components are designed to be independent.

REFERENCES

[1] Iqbaldeep Kaur, Parvinder S. Sandhu, Hardeep Singh and Vandana Saini, Analytical Study of Component Based Software Engineering – World Academy of Science, Engineering and Technology 50 2009. p.p 437-438.

[2] Herzum Peter, SIMS Oliver, Business Component Factory, Wiley 1999.

[3] Er. Iqbaldeep Kaur, Dr. P. K. Suri, Er. Amit Varma, Characterization and Architecture of Component Based Models International Journal of Advanced Computer Science and Applications Volume 1 –No.6, Dec 2010.p.p 66-68

[4] Arvinder Kaur and Kulvinder Singh Mann, Component Based Software Engineering – International Journal of Computer Applications (0975-8887) Volume 2, May 2010.p.p105-106.

[5] Arvinder Kaur and Kulvinder Singh Mann Component Selection for Component based Software Engineering, International Journal of Computer Applications (0975 – 8887) Volume 2 – No.1, May 2010. p.p 110-111.

[6] Jyrki Kontio, OTSO: A Systematic Process for Reusable Software Component Selection Dec 1995.

[7] STOJANOVIC Zoran, An Integrated Component-Oriented Framework for Effective and Flexible Enterprise Distributed SystemsDevelopment, http://www.betade.tudelft.nl/publications/ Stojanovic_*CAISE2002.pdf*, last access: 01.12.2005.

[8] DOGRU Ali H., TANIK Murat M., A Process Model for Component-Oriented Software Engineering, IEEE Software, March/April 2003. *p.p 35-39*