International Journal of Information & Computation Technology.
ISSN 0974-2239 Volume 4, Number 6 (2014), pp. 571-576

http://www. irphouse.com

Component Based Software's: Issues Related to Test the Compatibility of the Components

Damini Yadav[1] and Jagdeep Kaur[2]

[1]CSE-M.TECH, ITM University,
Village- Sanpaka, P.O- Pataudi, Dist-Gurgaon, Haryana, INDIA
[2]ITM University, Department of CSE/IT, ITM University, Gurgaon, INDIA

ABSTRACT

Component-based software development is an emerging new phase in software development because complex software's are developed from the integration of reusable components instead of developing everything from the very beginning each time. Component-based development is however still not mature process and there still exist many problems such as its maintenance, integration, testing and its compatibility with the existing system and so on. Often, components selected by functional features are incompatible or the integration effort required is too high. So doing the selection of components based on compatibility can simplify the integration task. Therefore in this paper, we discuss the various issues related to test the compatibility of the components.

Keywords:- Commercial off the shelf(COTS), Compatibility issues

1. INTRODUCTION

Component Based Software Engineering (CBSE) researchers over the last decades have created many tools and techniques that facilitate assembling large and complex systems from independent components. Despite their many advantages, these technologies also tend to push many problems and complexities into configuration and integration activities, [1]. Each component considers as black boxes, connected through links carry out some functionality but hide how they implement that functionality. These functionalities work as the base to build interconnections among components. Components as black boxes expose Ports which are used as connection points at a component's boundary; such ports are generally called Module Interconnection Languages (MIL) or Interface Definition Languages (IDL). Links are the association among the components' ports, [2]. The purpose of ports and links is to

enable components to co-operate to produce a desired joint functionality. Each component has multiple versions and there are complex dependencies between components and their different versions. Developers of component-based face many problems as first, it is very difficult to test all possible configurations of such a large number of components and their versions. Second, the components and their dependencies can change without notice. Commercially off the Shelf (COTS) components are integrated into the software products to reduce development time and effort. Even if extensively tested in isolation and in several contexts, system integrators must re-test COTS components in the specific contexts to look for possible integration faults not revealed in previous contexts. In practice, developers often approach this problem by conducting compatibility testing. So, component's compatibility can be checked and developers can verify components that match their requirements, functionality attributes prior of integrating them into their system.

2. OBJECTIVE

The objective of Compatibility Testing is to satisfy the consistent and relevant system requirement with high quality and reliability components for software development. Component development will lead to the final components with the help of the Compatibility testing which will select and test the candidate components that satisfy the requirements with correct and expected results. The system architecture are design to collect the users requirement, identify the system specification, select appropriate system architecture, and determine the implementation details such as platform, programming languages, etc.

3. STEPS

The different steps involved in the development process of component-based systems are, [3]:

- Find components which may be used in the system
- Select the components which meet the requirements of the system
- Alternatively, create a proprietary component to be used in the system
- Adapt the selected components so that they suit the existing component model or requirement specification
- Compose and deploy the components using a framework for components
- Replace earlier with later versions of components.

4. VARIOUS ISSUES

4.1 MATERIALS AND METHODS

Many organizations and companies are spending much time in reusable component selection since the choice of the appropriate components has a major impact on the project and resulting product, [4]. The component selection process is not defined so each project finds its own approach to it. Here a proper method should be introduce which supports the search, evolution and selection of reusable software and provides

specific techniques for defining the evolution criteria, comparing the cost and benefits of alternatives and consolidating the evolution results in decision making.

4.2 THE PHYSICAL PROPERTIES OF SOFTWARE COMPONENETS

It is commonly known that when building new systems, software developers are concerned with two main dimensions:

- Customer requirements: Functional, Non-functional
- System requirements: Syntactic, Semantic

These are the two different types of components matching: behavioral and structural matching, [5]. The ideal scenario where a component can be considered as an exact match to the requirements of the system is achieving both types of matching. For a component to be fully operational one satisfying the behavioral match is not enough, hence some work should be done in verifying the characteristics of the structural match. This structure of a component is named as its *physical properties* and these are define as the set of characteristics that a component must satisfy in order to match the structure imposed by a system. These characteristics constitute part of component's interfaces which are distinct in nature and different from one structural type to another. As a result, it is useful to utilize the characteristics defined by components physical properties to establish the basis for verifying component compatibility to system requirement. Various Physical Properties are shown in figure 1.

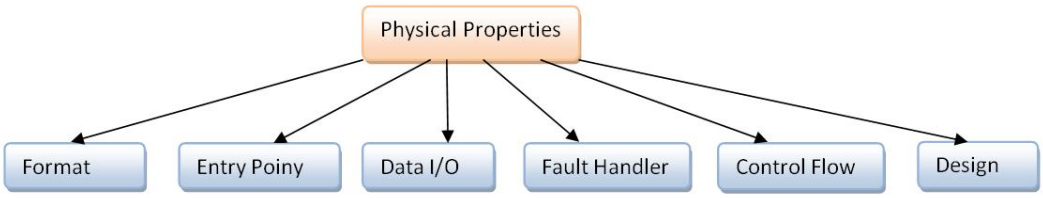

Figure 1.*Physical Properties of Componenets*

4.3 NEW VERSIONS

To keep their products competitive in the market vendors frequently release new versions of popular components. Consequently, software engineers often update COTS components integrated in their systems with either new versions of the same products or equivalent products, to keep the overall system up-to-date and competitive, [6]. After each update, test designers must design and execute regression test suites, to check that no new faults should be introduced in the system and verify its compatibility with the existing system. Checking the compatibility of all candidates may be extremely time consuming. So work should be done to tackle both the problem of generating efficient regression testing for compatible COTS components and compatibility testing among alternative components. Figure 2, shows generation of prioritized regression and compatibility test suites.

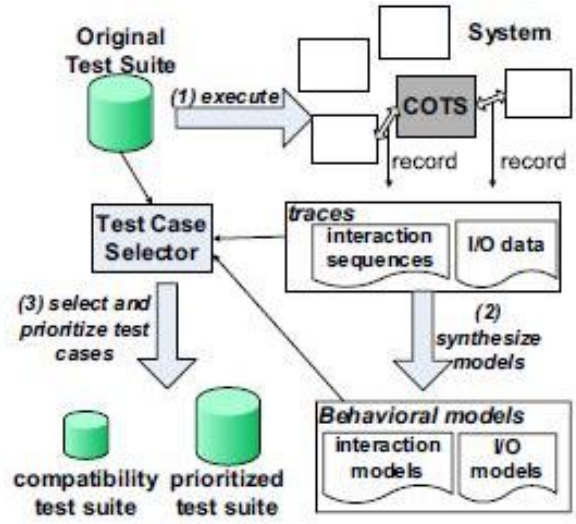

Figure 2. *Generation of Prioritized Regression and Compatibility Test Suites*

4.4 ROLES AND COMPATIBILITY PATTERNS

A "role" represents the possibility for components involved in the interaction to perform different, complementary responsibilities. Roles are a natural concept in patterns, architectures, architectural styles, etc. Where two or more components interact with each other is refer as "compatibility pattern". It is responsibility of the compatibility check mechanisms to determine which component is actually going to interact, based on the patterns their respective suppliers have specified. Hence, the role of the components should be specified properly for better interaction and compatibility between components. The various compatibility patterns discuss so far are shown in figure 3.

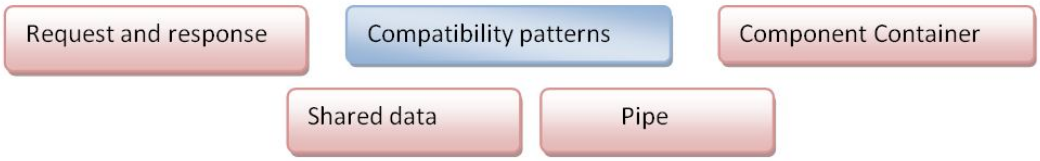

Figure 3.*Compatibility Patterns*

4.5 NETWORK EXTERNALITIES

Network externalities are the effects on a user of a product or service of others using the same or compatible products or service. Positive network externalities arise when customers value a product more if it is compatible with other consumers' products. In markets with network externalities compatible designs may emerge as the preferred choice by producers, [7]. Network externalities are a source of scale economies that arise from the demand side of the market. Hence, a compatible technological design

should be adopted to increase the value that consumers derive from a firm's product. Compatibility gives the benefits of other firms' networks to the consumers.

5. CONCLUSION

It is necessary to focus on each of the issues discuss above in order to understand and determine whether a newly designed component will physically fit and function within an already built system. A developer must first identify the characteristics and their values precisely, in order to ensure that the proper code and architectural interface can be used to verify software components before the integration. So the main consideration is given for providing the notion of architectural interface to verify a wider range of physical properties of components written in different programming languages. For successful component integration at the source code level without any compatibility issues we need the *designing of a tool* to check automatically the availability of the characteristics in the structural interface of software components.

6.REFERENCES

[1] Il-Chul Yoon, Alan Sussman, Atif Memon, Adam Porter, " Direct-Dependency-based Software Compatibility Testing", 2007.

[2] Alberto Sillitti, Giampiero Granatella, Paolo Predonzani, Tullio Vernazza, "Dealing With Software ComponentsCompatibility".

[3] Ivica Crnkovi´c, article in a regular journal, Journal of Computing and Information Technology, " Component-Based Software Engineering—New Challenges in Software Development", 2003.

[4] Muhammad Osama Khan, Ahmed Mateen, Ahsan Raza Sattar in American Journal Of Software Engineering and Applications, "Optimal performance model investigation in component-based software engineering (CBSE)", 2013.

[5] Basem Y. Alkazemi, Department of Computer Science, Umm Al-Qura University, Makkah, Saudi Arabia appeared in International Journal of Software Engineering & Applications (IJSEA), "On Verification Of Software Components", 1998.

[6] Leonardo Mariani, Sofia Papagiannakis and Mauro Pezz`e in 29th International Conference on Software Engineering, IEEE, " Compatibility and regression testing of COTS-component-based software", 2007.

[7] Giancarlo Succi, Andrea Valerio, Tullio Vernazza, Gianpiero Succi Appeared in ACM StandardView, "Compatibility, Standards and Software Production", 1998.