International Journal of Scientific Research in Computer Science and Engineering
Technical Paper Vol-1, Issue-6 ISSN: 2320-7639

CCTF: Component Certification & Trust Framework

Latika Kharb

Department of IT, JIMS, Sector-0, Rohini, New Delhi, INDIA, lkharb@gmail.com

Available online at www.isroset.org

Received: 28 Nov 2013 Revised: 08 Dec 2013 Accepted: 20 Dec 2013 Published: 30 Dec 2013

Abstract**—** Trust is important while assembling the components, as users want to know that a component will function as required and "advertised" and certification is a mechanism through which trust is gained. When both these notions are applied to component software development, we recognize that what is required to produce reliable and stable software, worthy of certification of certification of trust. We begin with defining key terms used in this paper and follow this with a discussion on trust and its relationship to certification. We continue with descriptions of a basic form of proposed CCTF: Component Certification & Trust Framework and then present an extension to basic framework that supports establishing trust.

***Index Terms* –** Trusted Component, Third-Party Certification, Component Based Software Project Manager

I. INTRODUCTION

Component-based software engineering (CBSE) / Component-Based Development (CBD) / Software Componentry is a branch of the software engineering discipline, with emphasis on decomposition of the engineered systems into functional or logical components with well-defined interfaces used for communication across the components [1]. A component should be able to be developed, acquired and incorporated into the system and composed with other components independently in time and space [2]. As a component is typically developed in a system environment that is different from environment of final system where component is integrated, so it's difficult to predict component behavior in a new system [3]. In this paper, we describe the framework, and describe two possible forms that the framework may take in order to establish trust among participants in component-based design.

II. TRUST AND CERTIFICATION

Establishing trust in a piece of hardware, software or a network is somewhat similar to the process that an individual uses to trust a service organization. Certification is a mechanism by which trust is gained and associated with certification is a higher level of trust that can be assumed when using implicit trust mechanisms.

Component users want to know that a component will function as "advertised". However, in this paper we restrict our scope to the framework's support for trusted interactions. At a minimum, there are five roles required to support component-based development of systems. In the following discussion, we describe each of these roles and follow up with a discussion of the ways in which the participants interact. We then discuss alternative mechanisms for extending the basic framework to address the issue of trusted component property values.

III. COMPONENTS OF THE CCTF

Components play a critical role in many software systems. Thus, our ability to reason about the properties of assemblies of components is of great concern to modern system implementers. Our ability to understand the functional and extra-functional properties of such systems suffer from:

- a lack of information about component behavior,
- a lack of confidence in the information that is available, and
- an inability to determine properties of component assemblies based on "black box" component representations.

In this section, we'll present an overview of component terminology used in this paper.

- Component Technology Specifier defines what it means to be a component as well as the types of interactions used to connect components.
- Component technology implementer provides the infrastructure that enforces the standards imposed by a component technology.
- Reasoning framework developer creates analysis techniques for predicting quality attributes of component assemblies.
- Component implementer creates components that are conformant with some component standard.
- System implementer assembles components to fulfill some high-level function.
- Component technology consists of a standard for developing and modeling components and a language in which to specify component assemblies.
- Component framework is a conformant implementation of a component technology.

Corresponding Author Latika Kharb

IV. REQUIREMENTS FOR A TRUSTED COMPONENT

A component should be able to be developed, acquired and incorporated into the system and composed with other components independently in time and space [2]. Certification certifies that it will do precisely this (for all contexts where its dependencies are satisfied). It will therefore provide a basis for component certification. Requirements necessary for a trusted component include:

- The CSPM (Component Software Project Manager) of the project and an engineering trained leader who motivates all members of the project team towards implementing the one best way.
- Verification by the sub-CSPM and third-party certification organization that the developers are adhering to design decisions of the project and subproject groups.
- Verification that performance specifications are followed, gaps identified, risks mitigated and test cases developed early during the design phase and validated at the completion of the project.
- Certification is a necessary activity to enable reasoning about the behavior of component-based systems. It is admirable that developers would be entrusted with considerable knowledge of the development of components.

V. INTERACTION AMONGST COMPONENTS IN CCTF

Typically, the development of component-based systems starts with a collection of existing components [4]. In Figure 1, we've shown the five basic elements of a component technology along with their interactions.

The semantics is described below.

1. Step 1: The component technology implementer uses the specification provided by a Specifier of the component technology to build conformant component infrastructure.
2. Step 2: The component technology specifier may suggest that a particular analysis technique be developed to support reasoning about some quality attribute that is important to the types of systems likely to be developed using this framework.
3. Step 3: The component technology implementer may suggest new types of analyses, and the developer of the reasoning framework will provide input as to what types of analyses are useful within certain types of systems.
4. Step 4: The component infrastructure provider must provide the system implementer with trusted infrastructure properties.
5. Step 5: The component implementer receives its standards information from the implementer of the component technology.
6. Step 6: The developer of the reasoning framework defines component properties that must be trusted, while the organization that actually build the component may be required to divulge otherwise hidden implementation details to support a particular analysis technique.
7.

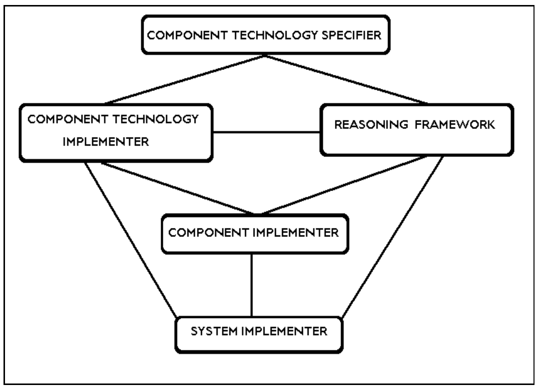
COMPONENT TECHNOLOGY SPECIFIER
COMPONENT TECHNOLOGY IMPLEMENTER
REASONING FRAMEWORK
COMPONENT IMPLEMENTER
SYSTEM IMPLEMENTER

Figure 1: The CCTF

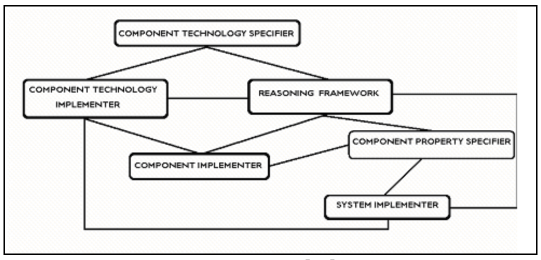
COMPONENT TECHNOLOGY SPECIFIER
COMPONENT TECHNOLOGY IMPLEMENTER
REASONING FRAMEWORK
COMPONENT PROPERTY SPECIFIER
COMPONENT IMPLEMENTER
SYSTEM IMPLEMENTER

Figure 2: Extended CCTF

8. Step 7: The developer of the system receives components and property documentation from the component implementer, and may notify the component implementer of the desire to use an analysis technique for which a component under consideration has not been validated.
9. Step 8: A reasoning framework developer supplies the system implementer with the necessary algorithms for performing the compositional analysis techniques, and the system implementer may indicate to the developer of the reasoning framework that a new type of analysis would be useful.

The five basic roles of Components are:

- Component technology Specifier defines what it means to be a component as well as the types of interactions used to connect components. The resulting specification defines a standard that must be adhered to by component and infrastructure developers.

- Component technology implementer provides the infrastructure that enforces the standards imposed by a component technology.
- Reasoning framework developer creates analysis techniques for predicting quality attributes of component assemblies. The developer of the reasoning framework defines what needs to be known about a component in order to be amenable to the technique. It may also supply the means by which to determine the component property.
- Component implementer creates components that are conformant with some component standard.
- System implementer assembles components to fulfill some high-level function. The system implementer expects to be able to predict the quality attributes of a proposed design before commitments are made to use specific components.

In Figure 2, we've shown our original framework extended with the addition of a Component Property Certifier.

- Component Property Certifier acts as a trusted third party. A property certifier might verify credentials that were provided by the component implementer along with components or, alternatively, create an additional credential. The component implementer trusts the certification organization with the source code for a component and the system implementer trusts that the certificates supplied along with components and obtained through a certifier are valid.

The original communication between the component implementer and the system implementer has been replaced by an interaction between the component implementer and the component property certifier and two new two-way interactions have been added.

- The redirection of Step 9 forces the supply of components to go through the certifier.
- The two-way Step 10 between reasoning framework developer and property certifier represents the mutually informing relationship between the developer of compositional analysis techniques and the user of them. The reasoning framework developer might devise an algorithm that requires knowledge of component internals that are not certifiable. In this case, the property certifier needs to notify the developer of the analysis technique so that adaptations to the algorithm can be explored.
- The two-way Step 11 between the property certifier and system implementer indicates that a request from the developer for components with specific certificates and property certifier's supply of those components.

VI. CONCLUSION

In this paper, we conclude that COSD is a promising discipline to be considered as one of the alternatives to promote software evolution and also to improve the software development process of currently complex systems with the help of our proposed CCTF: Component Certification & Trust Framework. Research and practice in the areas of component trust and certification, component technology, and software architecture to date has been conducted largely in isolation and has only touched on a few core issues.

REFERENCES

[1]. Paul Clements, From subroutines to subsystems: Component-Based Software Development, Allen Brown, Ed., Component-Based Software Engineering: Selected Papers from Software Institute, pages 3-6, 1996

[2]. Szyperski, C. Component Software: Beyond Object-Oriented Programming, Addison-Wesley, 1998. 213.

[3]. Kharb L. Assessment of Component Criticality with Proposed Metrics, IndiaCom 2008, BVICAM.

[4]. McInnis, Component-Based Development: The Concepts, Technology and Methodology, Castek Software Factory Inc., www.CBD~HQ.com

AUTHORS PROFILE

Dr. Latika Kharb is working in Department of MCA as GGSIPU Faculty in JIMS: Jagan Institute of Management Studies, New Delhi. She is an MCA, PhD(Computer Science). She has published over 30 research papers/articles in various International, National Journals and Conferences.