5th ICSE Workshop on Component-Based Software Engineering: *Benchmarks for Predictable Assembly*

Ivica Crnkovic[1], Heinz Schmidt[2], Judith Stafford[3], Kurt Wallnau[3]

[1]Mälardalen University, Department of Computer Engineering, Sweden, ivica.crnkovic@mdh.se
[2]Monash University, Australia, Heinz.Schmidt@csse.monash.edu.au
[3]Software Engineering Institute, Carnegie Mellon University, USA, {jas, kcw}@sei.cmu.edu

Abstract

This paper gives a short overview of the 5th ICSE Workshop on Component-based Software Engineering held at 24th International Conference on Software Engineering. The workshop brought together researchers and practitioners from three communities: component technology, software architecture, and software certification. The primary goal of the workshop was to continue clarifying the concepts, identifying the main challenges and findings of predictable assembly of certifiable software components. To focus the workshop on the topic, a call for papers was accompanied by a white paper, which provided a framework for invited papers and the workshop itself. The paper gives a comprehensive summary of the position papers, of the workshop, its findings and its results.

1 Introduction

The fifth CBSE (CBSE5) workshop held at the 24th International Conference of Software Engineering (ICSE) is a direct continuation of the fourth CBSE workshop (CBSE4) [1,2,3]. CBSE4 focused on reasoning about properties of assemblies from properties of components and their interactions. Researchers from three communities: component technology, software architecture, and software certification, joined the workshop, resulting in lively discussion and increased understanding of how the domains can be mutually informing. The need for a model problem, to be utilized for further research of different aspects of predictable assembly, was identified. The specification of model problems was discussed at a follow-up workshop held at the Carnegie Mellon University's Software Engineering Institute in Pittsburgh, U.S.A. The objectives of CBSE5 were defined at the SEI workshop.

The aim of CBSE5 was to more deeply study the problem of predictable assembly, focusing on the sub-problem of compositional reasoning, and benchmarks of the effectiveness of compositional reasoning. Submitters were asked to address the community model problem, either directly or indirectly by adopting the vocabulary of its specification.

This rest of this paper is organized as follows: Section two gives an overview of the workshop purpose and goal. Section three describes the workshop sessions. The paper concludes with description of future plans.

2 The Aim of the Workshop

The premise of the CBSE workshop series is that the long-term success of component-based development depends on the ability to predict the quality of component-based systems; however, developers are currently unable to make such predictions. Further research is needed in the area of predictable assembly to develop a component composition theory for reasoning about both the functional and extra-functional properties of component assemblies based on the properties of components.

Issues related to developing a composition theory include determining what properties are of interest to developers and users of components, how to predict the properties of assemblies, how to measure properties of components, how to verify the measurements, and how to communicate the property values to component users. Resolving these issues requires collaborative work of researchers in several domains including compositional reasoning, composition languages, component trust and certification, software architecture, and software components.

2.1 Workshop Objectives

The solution to the problem of predictable assembly is the identification and application of a component composition theory, which is based upon both constructive and analytic techniques. The primary goal of CBSE5 was to achieve a better understanding of compositional reasoning techniques and to test the feasibility of their use through their application to community model problems.

A composition theory assumes the availability of information about the properties of components but, in practice, there is no established method for measuring and communicating this information. Thus, a secondary goal of the workshop was to raise issues related to understanding how to provide this information. Examples include identification and formal specification of properties that convey information about component internals, measurement techniques for assessing the properties, methods for certifying these measurements, and methods for communicating the resulting values.

To make the workshop efficient, it was essential to obtain a clear understanding of what constitutes a problem of predictable assembly, and what qualities attend to their solution. A guide for model solutions was available in the form of a white paper [4] to prospective workshop participants in which the authors presented a structure and vocabulary to serve as a basis for the clear understanding. The white paper outlines the content of a hypothetical report of a model problem and its solution. It also introduces a vocabulary of predictable assembly. We quote an essential definition offered by the white paper:

> *A problem in predictable assembly is characterized as a software engineering problem that can be reduced to the form: Given a set of components C, predict property P of an assembly A of these components. At the core, a solution to such a problem involves a prediction theory that is based on certain assumptions about the environment in which the assembly will run and requires information about the components that make up the assembly, thus there are many peripheral issues that reside within the bounds of research in predictable assembly.*

CBSE5 was thematically centered on this definition of predictable assembly.

2.2 Workshop Organization

The workshop involved more discussion than paper presentations. For this reason the authors of all papers made very brief presentations of their work (3 minutes per paper!). After that the three papers were selected by voting for presentation in detail.

Based on the presentations and related discussion, the workshop continued with breakout groups focused on the following topics:

?? Component properties and emergent properties;

?? Component and system reliability; and

?? Component Containers

The results of these breakout discussions are summarized in Section 4.

3 Participating in the Workshop

Attendance at the workshop was by invitation, in large part, based on acceptance of position papers. Submitters were asked to:

- clearly state the problem area;
- provide an overview of the domain by way of background;
- describe a family of components associated with the problem;
- state properties that developers want answered about an assembly;
- detail a technique for reasoning about the property;
- validate or at least discuss plausibility of validation of the technique; and
- detail the reasoning technique by way of example.

The following position papers were accepted for the workshop (the abstracts are presented here, for the full papers see [5]):

Gary Vecellio, William M. Thomas, and Rob Sanders, **Containers for Predictable Behavior of Component-based Software**

Component developers have limited knowledge of how their components will be aggregated into applications and they can not control the deployment and execution environment. This makes the development of predictable component-based software a difficult proposition. Adding services to a software container can help remedy this problem. This paper discusses how commercial container technology can be augmented to support more predictable behavior of component compositions. Our approach consists of augmenting an open source Enterprise JavaBeans container and server with assertion capabilities. We discuss how these new capabilities can be used at load and initialization time to verify that a composition meets some policy constraints and at runtime to verify that the composition is maintaining critical properties.

Paola Inverardi and Massimo Tivoli, **Correct and automatic assembly of COTS components: an architectural approach**

Many software projects are base on the integration of independently designed software components that are acquired on the marker rather than developed within the project itself. This type of components is well known COTS components. Nowadays component-based technologies COM/DCOM, Sun’s JavaBeans, CORBA) provide interoperability and composition mechanisms that cannot solve COTS component assembling problem in an automatic way. Notably, in the context of component-based concurrent systems, the COTS component integration may cause deadlocks or other software anomalies within the system. In this position paper, we present our approach to contribute top the research in components assembly. Our long term goal is to develop a tool that synthesis the assembling code to glue together a set of COTS components. This glue code must be synthesized in such a way that (as well as defined set of) functional properties required for the composed system are automatically guaranteed. We propose an architectural connector-based approach for the assembly problem. The basic idea is to build applications by assuming a defined architectural style. Then, we compose a system in such a way that it is possible to check whether and why the system presents some anomalies (.e.g. deadlock, livelock). Based on the analysis results a recovery policy which can avoid the anomalies and obtain a correct assembly can be performed.

Judith A. Stafford and John D. McGregor, **Issues in Predicting the Reliability of Composed Components**

Availability is one of the most frequently specified quality attributes for computerized systems and the computation of availability requires knowledge about the reliability of the system. Although much research has been devoted to software system reliability, much work remains to be done in identifying ways to predict reliability of assemblies of components. We are designing an experiment for use as a foundation for creating a reliability prediction-enabled component technology (PECT), which is to be used to produce systems that are predictably reliable by construction; in the course of that work we have recognized the need to evolve combinatorial reliability models for use in computing reliability of assemblies based on the reliabilities of constituent components. In this paper, we describe and discuss aspects of current models that need to be adapted and how they affect the design of our experiment.

Jason O. Hallstrom, Scott M. Pike, and Nigamanth Sridhar, **Iterators Reconsidered**

Software developers are eager to increase the scale of their software products at a rate proportional to the growth of computing resources. With memory, bandwidth, and computing power doubling roughly every eighteen months, development approaches that are not based on compositional reasoning techniques can not be used to engineer the systems of tomorrow. The enormous scale of these projects far outstrips our ability to understand them using ad-hoc approaches. Industry best practice recognizes the importance of component reuse, but the emphasis is weighted heavily on the reuse of component code, often times neglecting the need to reuse the effort that went into understanding the component's behavior. That is, any scalable software engineering discipline must provide mechanisms for reusing software components, as well as mechanisms for reusing the reasoning effort required to use those components. This paper examines the Iterator pattern with regard to compositional reasoning. The approach, touted as industry best practice, is shown to provide sample opportunity for breaking the principles of encapsulation. These various hazards are briefly described, and several techniques for ensuring safe use of the pattern are explored.

Heinz W. Schmidt and Ralf Reussner, **Parameterized Contracts and Adapter Synthesis**

Ideal reuse takes a component as it is. However, ideal reuse is a myth and hardly achieved in practice. More commonly the software architect modifies, adapts and reconfigures components. Sometimes complex synchronizations between several components are necessary before they can be deployed.

This paper presents some recent results and work in progress on component adaptation. We develop methods for identifying incompatibilities and automatic synthesis of adapters which control components dependent on their deployment context. Typically such adapters are considerably smaller than the components themselves. They belong into the realm of the connectors.

The synthesis of adapters requires compositional reasoning about extra-functional aspects of component and system behavior such as the order and timing of events, the possible matches and mismatches of such orders or partial orders. Extensions of our work currently in progress include probabilistic information associated with behavior specifications capturing usage profiles or reliability information.

Using concrete examples we show adapter generation for mismatching protocols including generation synchronizing controllers for shared resources. We only sketch our extensions to deal with reliability.

Shiping Chen, Ian Gorton, Anna Liu, and Yan Liu, **Performance Prediction of COTS Component-based Enterprise Applications**

One of the major problems in building large-scale enterprise systems is anticipating the performance of the eventual solution before it has been built. This problem is especially germane to modern Internet-based e-business applications, where failure to provide high performance and scalability can lead to application and business failure. The fundamental software engineering problem is compounded by many factors, including application diversity, architectural trade-offs and options, COTS component integration requirements, and differences in performance of various software and hardware infrastructures. This paper investigates the feasibility of providing a novel and practical solution to this problem. The approach as demonstrated, constructs useful models that act as predictors of the performance for component-based systems hosted by middleware infrastructures such as CORBA, COM+ and J2EE.

Dave Mason, **Probabilistic Analysis for Component Reliability Composition**

One of the desirable properties of predictable assembly is reliability. Given reliability and transformation functions for components, it is possible to accurately compose reliabilities. Currently the transformations are limited in their domain of applicability, but we are working to extend their domain.

Chang Liu and Debra J. Richardson, Specifying **Component Method Properties for Component State Recovery in RAIC**

Redundant Arrays of Independent Components (RAIC) is a technology that uses groups of similar or identical distributed components to provide reliable services to applications. RAIC controllers use the just-in-time component testing technique to detect component failures. RAIC also allows components in a redundant array to be added or removed dynamically at run-time. Component state recovery techniques are used to bring replacement components or newly added components up-to-date. Two types of state recovery techniques are used in RAIC: a snapshot-based approach and an invocation-history-based approach. Component method properties are used to optimize invocation-history-based component state recovery. This position paper gives a brief overview of RAIC and discusses the component state recovery techniques used in RAIC. A proof-of-concept example is given to illustrate how a problem occurs in a component is detected and how a replacement component is brought up-to-date automatically to substitute the fail component.

Gabriel Moreno, Scott Hissam, and Kurt Wallnau, **Statistical Models for Empirical Component Properties and Assembly-Level Property Predictions: Toward Standard Labeling**

One risk inherent in the use of software components has been that the behavior of assemblies of components is discovered only after their integration. The objective of our work is to enable designers to use known (and certified) component properties as parameters to models that can be used to predict assembly-level properties. Our concern in this paper is with empirical component properties and compositional reasoning, rather than formal properties and reasoning. Empirical component properties must be measured; assessing the effectiveness of predictions based on these properties also involves measurement. This, in turn, introduces systematic and random measurement error. As a consequence, statistical models are needed to describe empirical component properties and predictions. In this position paper, we identify the statistical models that we have found useful in our research, and which we believe can form a basis for standard industry labels for component properties and prediction theories.

Nazareno Aguirre and Tom Maibaum, **A Temporal Logic Approach to Component-Based System Specification and Reasoning**

We propose a language for component-based system specification and reasoning. This language provides a new coarse-grained unit of modularization, which, we believe, allows one to better organize a system specification, and which admits the definition of (dynamic) reconfiguration operations. The language is mainly based on temporal logic as a formalism to describe behavior. Temporal logic is used to specify both internal behavior of components and architectural aspects of a system. This provides a uniform framework to reason about systems, allowing one to combine properties of components and architectural properties in a convenient way, even in cases in which the architecture could change over time. Some constructs provided by the language can be used to organize specification in a hierarchical way, which is more suitable for reasoning. The use of temporal logic provides an expressive language for stating properties. The powerful proof calculus associated with the language allows us to prove properties effectively, taking advantage of the structure of the specification.

Magnus Larsson, Anders Wall, Christer Norström, and Ivica Crnkovic, **Using Prediction Enabled Technologies for Embedded Product Line Architectures**

Predicting the behavior of a product before it is built has been a long time struggle, especially for software based systems. For building software systems there are few methods that comply with the engineering methods established from physics where properties of a construction can be determined before the actual assembly of a product. By taking the predictable assembly from certifiable components (PACC) approach our intention is to define methods to predict certain properties. We conclude that product line architectures that build on top of a component technology can be built in a much more controlled way if the component technology is prediction enabled. The aim of this position paper is to investigate how embedded product line architectures can utilize a prediction-enabled component technology to build products with known properties. We present a framework where we can reason about extra-functional properties in a uniformed way. We illustrate our approach by an example including the properties end-to-end deadline and version consistent.

4 Workshop Results and Future Plans

The breakout discussions focused on issues pertaining to reliability, compositional reasoning, and the role (and meaning) of "containers" in software component technology. The working group reports and closing discussion mirrored this breakout structure, with the exception that the topic of reliability was broadened to questions about the meaning of properties, and the meaning of "emergence." These highlights of these discussions are briefly summarized.

4.1 Component and Emergent Properties

Three interesting points were made during this discussion:

1. Property theories, for example those concerning time, are often in the form of either boundary case or average case predictions. Boundary conditions are often simple and easy to verify; on the other hand, boundary conditions are very difficult to validate. Average case predictions are the converse: easily validated, but quite difficult to verify.

2. All properties of executing software are, in theory, predictable. This contrasts with the conventional meaning of "emergence" as equivalent to "unpredictable." Instead, emergence occurs with respect to a particular (set of) property theory(ies). An observable property that is not predicted by a theory is emergent with respect to that theory.

3. We expect the real challenges of predictability to arise as a result of non-orthogonality of property theories. For example, assume a theory of time and a theory of reliability, each based on a set of assumptions. These assumptions, when combined, may yield emergent properties. A new combined model that captures this emergence is possible in theory, but may introduce arbitrary complexity.

4.2 Component Containers

The term 'container' or, alternatively, 'component container' has recently been associated with software component technology. The question posed was whether 'container' is a new concept, and, regardless, what role does it play in predictable assembly?

Several answers were proposed, among which the following were the most interesting, only a few of which require elaboration:

?? Containers are operating systems for software components, used for managing component life cycles and access to shared component resources.

?? Containers are virtual machines that define standards for component deployment and execution.

?? Containers are environments that can be used for formal reasoning, and for defining composition operators.

?? Containers are execution environments that provide services to component implementations.

?? Containers are scopes for properties. Just as a component defines a scope for properties (the component interface), an assembly of components has properties; this assembly exists within a container (which can be distributed).

In all of these cases, the general consensus was that containers, while not a fundamentally new concept, are not well understood with respect to their role in predictable assembly.

4.3 Composition

The discussion focused on models of composition. An assertion was made that a property theory, or reasoning technique, is compositional if and only if it has an algebraic model. In this view, component properties are the carriers of the algebra, and compositionality is expressed as operators (of arbitrary arity) over this carrier.

After some discussion, it was agreed that being algebraic is sufficient, but not necessary, for compositionality. Examples were cited where compositional reasoning is possible but where no pre-defined algebra is possible, for example, performance models that require as input an entire assembly or topology. In such cases reasoning is still compositional in that it follows from component properties that are expressed on the interfaces (boundaries) of components, but the compositional operator is unique to the assembly.

There was some discussion of whether compositional reasoning is well understood by someone, perhaps not by the workshop participants, but by researchers in other areas of computer science, for example, formal verification, software architecture, process algebras, etc. While it was not be possible even in principle to confirm this possibility, the participants did agree that there was a diversity of opinion in the workshop as to the nature of compositionality.

4.4 Publication of Results

The proceedings of the workshop are available on the web [2]. The proceedings and results of the workshop are also being used as a basis for a special issue of Journal of Systems and Software that is already announced [6].

5 References

[1] I. Crnkovic, H. Schmidt, J. Stafford, K. Wallnau, 4th ICSE Workshop on Component-Based Software Engineering: Component Certification and System Prediction, Software Engineering Notes, 2001. Nov

[2] http://www.sei.cmu.edu/pacc/CBSE4-Proceedings.html

[3] http://www.csse.monash.edu.au/dsse/CBSE4

[4] I. Crnkovic, H. Schmidt, J. Stfford, K. Wallnau, White Paper: Anatomy of a Research Project in Predictable Assembly, http://www.sei.cmu.edu/pacc/CBSE5

[5] http://www.sei.cmu.edu/pacc/CBSE5/CBSE5-Proceedings.html

[6] http://www.sei.cmu.edu/pacc/CBSE5/JSSCall.html