4th ICSE Workshop on Component-Based Software Engineering:
Component Certification and System Prediction

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 4th ICSE Workshop on Component-based Software Engineering. The workshop brought together researchers and practitioners from three communities: component technology, software architecture, and software certification. The goal of the workshop was to find a common understanding and to discuss the topics related to the component composition. The workshop was divided in eight sessions held in sequence, starting with invited talks and ended with a final discussion. A model problem, to be used for further research and work in future workshops, was discussed and later selected. The paper gives a comprehensive summary of the sessions and plans for future work.

1 Introduction

Although recognized as an emerging sub-discipline of Software Engineering in the late 90s, and widely accepted by researchers and industry communities, Component-based Software Engineering (CBSE) is still an immature discipline: not yet defined by researches and far away from fully explored by practitioners. CBSE continues to build its profile as is evidenced by its presence at many conferences of different types and the numbers of workshops that address CBSE from different points of view. The four CBSE workshops held at the International Conference of Software Engineering (ICSE) have similar objective: seeking the profile and boundaries of CBSE, and building the CBSE community. The 4th CBSE workshop focused on reasoning about properties of assemblies of component. 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. A model problem, to be utilized for further research of different aspects of CBSE, has been defined.

1.1 Previous CBSE Workshops

Since 1998, there have been four successive workshops on Component Based Software held in conjunction with the International Conference on Software Engineering. The first ICSE workshop on CBSE was held in Spring, 1998 in Kyoto, Japan [2]. This workshop was small, but attracted senior researchers from all points of the globe. The objectives of this gathering were to develop a synopsis of the current state of CBSE in research and practice, and to determine whether a community of CBSE research could be established under the auspices of the ICSE conferences. The results of the workshop are summarized in a short paper by Brown and Wallnau [1].

The second CBSE workshop was held at ICSE'99 in Los Angeles, USA [3]. This workshop was much larger and attracted researchers from a number of related but distinct software research areas, such as software reuse and software architecture. Over fifty attendees met for two days in five breakout groups discussing a variety of technical, process, and business issues. If anything, though, the workshop was too diverse.

The third CBSE workshop was held in Limerick, Ireland [4]. Partly as a response to the extreme diversity of viewpoint evident in Los Angeles, the theme of this workshop was a narrower reflection on practice, with workshop participants asked to respond in their position papers to several specific questions related to research and practice. One of the key results of this more focused workshop was a set of pressing research topics. Heading the list were issues of predictable assembly and component trust and certification. This led directly to the 4th ICSE workshop on CBSE.

1.2 Different aspects of CBSE

There are many areas covered by component-based software engineering. First, CBSE is concerned with the adaptation of methods from many areas software engineering for application to component and component-based development. Second, methods specific to component-based development are close related to other disciplines (for example, reuse, object-oriented development, software architecture, etc.). Many problems arising in CBSE have been discussed within other communities, and in some cases the communities were not aware of the results already achieved. For example, software architecture and software certification, which have been discussed in different communities, are extremely important topics in CBSE.

The software architecture defines the components of a software system as well as their interactions and can be used to analyze its quality attributes. The software architecting defines the components that make up the system; the properties and functionality of the components and their interactions define the overall system functionality and behavior.

Component specification is crucial for successful implementation of CBSE. This is also one of the most important open topics. While different component technologies have been successful the definition of application programming interface (API) specification (i.e. in specification of functional part of interface) is well understood, there is a lack of understanding of what and how to specify and verify extra-functional properties (also known as non-functional) that leak through the component's interface and affect overall system behavior and quality. Component certification is one way to make possible the use of components to build systems that behave in a predictable manner.

The workshop and the conference organizers have realized that many benefits can be achieved if the three communities, component technologies, software architecture, and software certification, meet together, share experiences and find common directions for future research. The main challenge of the workshop was to discover whether these three communities could find the common language, identify the problems, and determine directions for possible solutions?

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 aim of the workshop was to develop a shared understanding of certifiable component properties and predictable assembly of components. To achieve this the Call for Papers included the following statements:

Papers should relate to the workshop objectives:

– Define the problem space, for example software properties amenable to compositional analysis, measurement, and prediction.
– Specify one or more open problems, "grand challenges", or critical gaps that can provide a cynosure for community development.
– Relate the research activities of targeted workshop participants to community problems and identify collaboration potential.

Papers should state a position with respect to the following issues:

1. What do developers want to predict about component assemblies?
2. What compositional reasoning techniques are available to support prediction?
3. Which of these techniques benefit from knowledge of component internals and what do they need to know?
4. What can be known about component properties in the absence of knowledge of the context in which it will be deployed and used?
5. How do we measure those properties and what degree of precision is required?
6. How is this information made available by the component?

3 Organization of the Workshop

According to the topics of interest, the workshop was divided into eight sessions; six working sessions held in sequence between a welcome session and a closing session. In total 20 papers were selected and presented at the workshop. The opening session was very important since it had a goal to bring together the three communities, to find the common language, recognize the common problems and focus on the common goals. This session included introductions to the three focus areas: component trust and certification, component technology, software architecture. Three keynote speakers gave a short introduction to these areas. The six working sessions discussed the six issues listed in the Call for Papers. Each working session started with a short presentation by the session chair describing the issue and its relationship to the other five issues. The session continued with short presentations of the papers and concluded with a discussion related to the topics and to the papers. The presentations and discussions varied from session to session as it was up to each session chair to form the session. In this way a better dynamics and focus on more important aspects of the problem were achieved.

3.1 Keynote Speeches

3.1.1 Component Trust and Certification

Jeff Voas opened the session with a presentation outlining his view of component trust and certification. His position was: (1) Commercial software could be tagged with certificates that define minimal guarantees about how a software "unit" will behave in the future (and under what assumptions it will behave in those manners and (2) software vendors will never provide this. That is why software quality certification is needed. A software quality certificate is simply a fact sheet that spells out known software output behaviors and under what conditions these occur.

3.1.2 Component Technology

Clemens Szyperski motivated why component paradigm is attractive, and then analyzed the basic properties of software components (which are not the same as other type of components, such as specification components, architectural components, etc.), indicated the differences and similarities between components and objects or classes, as well as between components and modules. Finally he pointed to strong the strong relationship between component-oriented architecture and architecture-aligned components.

3.1.3 Software Architecture

David Garlan summarized the nature and purpose of work in the area of software architecture. He described issues addressed in an architectural design, presented an example architectural description, and reviewed several architecture description languages (ADLs). He described the elements of architectural description and explained why it is important to describe not only the structure of a system but also the behavior. He finished with a description of the Aura project that is currently underway at Carnegie Mellon University, which is concerned with the mobile computing environments.

3.2 Session 2: Relevant System Properties, Moderator George Heineman

In this session we focused on those relevant system properties that meet the following criteria:

– The component producer can include specific information regarding the desired property in the component's descriptive documentation.

– The component assembler can verify at design time, hopefully with tool assistance, that the resultant composition of components ensures the system property to an appropriate degree.

– Once the final system is complete, it must be possible to verify the existence of the desired property.

The session included three papers:

- S. Ghosh and A. Mathur, "Certification of Distributed Component Computing Middleware and Applications."
- D. Wile, "Ensuring General-purpose and Domain-specific Properties using Architectural Styles."
- J. Stafford and K. Wallnau, "Is Third-Party Certification Necessary?"

Sudipto Ghosh and Aditya Mathur discuss the issues in certifying applications built to the CORBA Component Model (CCM). There will certainly be a need to reconcile local certification, the verification that an individual component satisfies a specific property, with middleware certification. Because CORBA has many possible vendor implementations, it is possible that a specific CORBA application will guarantee a property with one vendor's implementation, but not another's.

David Wile raises the issue of validating a set of desired properties in concert with each other, rather than in isolation. He aims to identify composition principles for software architectural styles, a common theme from the software architecture community.

Judith Stafford and Kurt Wallnau propose that it may not be necessary to vest a single dedicated organization with the responsibility of certifying properties of components. In their model, the component itself is packaged within an "active component dossier" that defines the component credentials and provides test harnesses or benchmarking mechanisms to enable unbiased observers to verify these properties.

Activity of this session centered on discussion of the relationship between component properties and emergent properties of assemblies of components. The group discussed candidate properties to consider when reasoning about the composition of components into systems. To support compositional reasoning we must find properties that satisfy the equation $\rho_1(A?B) = ?(\rho_2(A), \rho_2(B))$. That is, we must understand the way components A and B are composed together (?) as well as how to specify property independently for A and B. There was general agreement that the property of composition might be based on other types of properties of the components. In other words, that ρ_1 and ρ_2. might be different types of properties. We agreed there is a need for continued research on defining properties.

3.3 Session 3: Properties of Separate Components, Moderator Betty Cheng

Reasoning about functional and extra-functional quality attributes of a component-based system generally involves knowledge of specific properties of the assembled components. Several questions come to mind when discussing properties of the individual components as well as properties resulting from their integration. For example, what can we know about the properties of a component when we do not have the context in which the component will be deployed and used? Some properties, such as end-to-end latency, require measurement in a test-harness type environment. Others, such as encryption strength, are properties of the algorithm used by the component. While others, such as potential input-to-output data and control pathways, must be identified from the source code of the component.

Papers presented in this session were:

- P. Mohagheghi and R. Conradi, "Experiences with Certification of Reusable Components in the GSN Project in Ericsson, Norway."
- P. Popov, L. Strigini, S. Riddle and A. Romanovsky, "Protective Wrapping of OTS Components."
- M. Woodman, O. Benediktsson, B. Lefever and F. Stallinger, "Issues of CBD Product Quality and Process Quality."
- D. Garlan and B. Schmerl, "Component-Based Software Engineering in Pervasive Computing Environments."

These four papers discuss a variety of aspects of component reuse. The papers describe what information needs to be known about components in order to facilitate CBSD, how to obtain that information, the impact of that information, how to encapsulate that information in terms of wrappers, and how to adapt components to changing environments. One paper also discusses the impact of the software development process on the reusability and composability of components. In this session we will explore these and other issues surrounding our ability to identify, analyze, and measure properties of components in isolation so that they can be composed in predictable, reusable, and useful ways. Three specific questions that we will attempt to address are as follows. How can we and should we certify reusable components (what are the criterion)? What properties of a component will maximize its reusability, composability, and adaptability? Which approach has the most potential benefits in terms of costs: domain-specific or domain-independent components?

3.4 Session 4: Compositional Reasoning – Moderator: Murali Sitaraman

Compositional (or modular) reasoning is fundamental for accomplishing the central workshop goal, i.e., predictable assembly of component-based systems. There is a near consensus on the meaning of compositional reasoning in the group: It is reasoning about the (functionality and performance) behavior of a system using the (functionality and performance) *specifications* of the components of the system, without a need to examine or otherwise analyze the implementations of those components. The ability to do *a priori* compositional reasoning is essential for engineered systems to work in predictable ways.

This session used a panel format. The following four papers are represented in this session:

- B.W. Weide, "Modular Regression Testing: Connections to Component-Based Software."
- T. Genßler and C. Zeidler, "Rule-Driven Component Composition for Embedded Systems."
- D. Mason, "Probability Density Functions in Program Analysis."
- H. Schmidt, "Trusted Components: Towards Automated Assembly with Predictable Properties."

Here follows a summary of the presentations and key topics of discussion.

The RESOLVE language and approach (presented by Weide) is intended for developing component-based systems with predictable behavior. In RESOLVE, all components have formal specifications that serve as contracts between developers and clients of components, in the sense of Meyer's design-by-contract principles. Highly parameterized component implementations can be written in RESOLVE programming language and can be verified to be correct in a modular fashion. A novel aspect of RESOLVE is the use of swapping as the basic data transfer mechanism for passing objects as parameters and for transferring values from one object to the other Swapping avoids aliasing, and thus permits "value-based" semantics in reasoning about objects while at the same time allowing references to be used in the underlying implementation for efficient data transfer.

The work of Schmidt's group focuses on trusted development and analysis of distributed systems. The group has studied contract-based approaches with a view to enrich common interface definition languages (IDLs) and architecture definitions with behavioral specifications to enable compositional reasoning about systems in-the-very-large. As in the case of RESOLVE, Schmidt's approach relies on the central idea that requirements to the (client) environment are explicitly stated to provide an explicit separation between interacting components. Component replacement in the environment must satisfy the requirements and hence, preserve the stability of the system that is already proved. Though the focus of this work is on component-based concurrent, real-time systems, Schmidt notes that most common contract-based approaches (e.g., Eiffel) rely on global analysis to establish system validity, making them non-compositional even in the sequential case.

The objective of PECOS, presented by Genßler, is to facilitate rule-driven component-based composition of embedded systems. PECOS is a Prolog-based prototype system and it emphasizes a "correct by construction" approach where possible. In embedded systems (the current focus of PECOS), the requirements are stringent and correctness is even more important than in other systems. Given the nature of the domain, PECOS does not support dynamic creation of component instances. While this allows for a number of static predictions about the behavior of the system, it does limit the class of systems that can be handled. However, this is not a serious limitation because embedded systems usually cannot tolerate dynamic component creation. This observation raised an interesting discussion on the extent to which predictability can be guaranteed in the presence of dynamic changes in componentization. At the very least, it appears that the scope of the changes would have to be bounded statically to ensure any measure of predictability.

All the three approaches discussed above involve investigation of techniques for predictable composition of non-functional behavior, with particular emphasis on time and space aspects of performance. Mason's work suggests that that it may not be possible to compose certain non-functional properties in a scalable fashion. Using probability distribution functions (PDFs) as examples, Mason's work explains how component code can be transformed into PDFs parameterized by arguments to the component. In this analysis, property specification of a component is exactly what is extracted from its code (and it is usually not abstract). Alternatively, the analysis of a component for certain properties depends on the internal details of every component it uses.

Given the background of Mason's work, one of the key discussion questions is what properties other than functionality might be amenable to compositional reasoning. A specific research question concerns whether it is possible to have simultaneously abstract yet precise specifications of performance behavior that make compositional performance reasoning possible. The difficulty and importance of this challenge for predictable engineering of component-based systems is further discussed in the summary of the session on Prediction and Measurement.

3.5 Session 5: Internals versus Abstraction, Moderator: Dave Wile

A primary reason component-based technologies are adopted is that reasoning about component behaviors can be raised to levels of abstraction above machine-, system-, or programming language- representations. A second useful abstraction lies in the definition hierarchy among components. This session covered three innovative approaches to abstract reasoning about component structure and behavior.

Papers presented in this session were:

- K. Fisler, S. Krishnamurthi and D. Batory, "Verifying Component-Based Collaboration Designs."
- D. Hamlet, "Component Synthesis Theory: The Problem of Scale."
- K. Lau ,"Component Certification and System Prediction: Is there a Role for Formality?"

Fisler, Krishnamurthi and Batory are concerned with the construction of systems as interacting layers or "collaborations" whose contribution to the whole is abstracted into features of the resulting system. Collaborations provide a composition technology quite orthogonal to conventional component decompositions. The roles each actor in the system plays in the various collaborations form the focus of their specification and verification technology.

Hamlet argues that the mere fact that components are used in truly large-scale systems changes the nature of the properties one wishes to prove and/or measure. In practice, substantially different abstraction mechanisms are used in large-component reasoning, e.g. average performance or worst-case analysis. This paper seeks a theory for making the connection between the macroscopic and microscopic views of components.

In his somewhat whimsically titled paper, Lau argues that in some sense component reasoning has been at too abstract a level, in that much information necessary for the use of a component is not revealed by the designers. Moreover, the abstraction process itself is often an after-the-fact activity. As distinguished from hardware components, software components are not designed to well-elaborated principles of design and semantic standards that manifest properties critical to a component's use in a real system.

These papers have many common threads despite little common terminology. Each takes a swipe at conventional abstraction techniques and illustrates how many problems with measuring, modifying, adapting, and using today's technology arise from our overly simplistic view of the nature of abstraction. Each proposes a unique approach to solving these problems and should stimulate lively discussion.

3.6 **Session 6: Measurement and Prediction –** Moderator: Dimitra Giannakopoulou

A number of issues need to be resolved before a component-based approach can make a significant impact to software development. Methods must be developed that allow *measurement* and *prediction* of the functional and non-functional characteristics of a system based on properties of system components. Component suppliers must be able to inform consumers about properties of components in a reliable fashion. What these properties are, whether they are context-independent, how they should be specified, and how precise measurements should be, are all open questions.

Three position paper presentations initiated discussions in this session:

- O. Preiss and A. Wegmann: “Towards a Composition Model Problem Based on IEC61850.”
- M. Sitaraman: “Compositional Performance Reasoning.”
- B. Councill: “Managing Software Component Processes.”

The substation automation domain was proposed by Otto Preiss as a model problem for research on component assemblies. The standard IEC61850 defines substation automation functionality based on collaborations of atomic functional units. This application domain provides concepts of system operations, including Quality Attribute (QA) requirements such as performance, reliability, and security. These requirements must be guaranteed before such systems are assembled. If functional units are realized as software components, assembling automation applications with specific QAs may be viewed as creating predictable component compositions.

Otto concluded with the following observation. Individual component properties are less of an issue when constructing predictable assemblies; rather, it is the infrastructure and the interactions between components that play the main role.

Murali Sitaraman explained that performance predictability refers to the ability to describe aspects of a system’s performance *before the fact*, as opposed to *observation* of performance on the final product. Techniques for performance prediction can only scale if they are compositional, i.e., if they reason about performance of the system based on performance characteristics of its components. However, performance predictability is inherently hard, which is to be expected if one considers that it is already hard to be exact about performance of basic components. This claim was substantiated by three examples.

Example 1: Assume a procedure *P*, which simply makes a call to another procedure *Q*, whose worst-case complexity is associated with parameter values that *P* never passes to *Q*. Then predicting the performance of *P* as that of *Q* is arbitrarily bad. Performance composition therefore requires *point-wise* performance specifications for reused components.

Example 2: How much time does it take to deep-copy a stack? For the result to be useful, one needs to factor in the values of objects that the stack contains, rather than simply its size. Performance specification (and reasoning) is therefore a meaningless exercise without behavioral specifications (and reasoning).

Heinz Schmidt observed here that results in the parallel computing field partly address the second problem. Oblivious algorithms (whose performance characteristics do not depend on values of objects handled) are distinguished from non-oblivious ones. A surprisingly large class of useful algorithms turns out fall within the former category.

Example 3: Suppose the abstract model of some container is “set”. Can we express the time taken to search this structure using the abstract model? Abstract models need to be augmented in order to specify performance precisely.

The speaker concluded that such issues are just the tip of the iceberg. In general, if we use parameterized components then performance specifications need to be parametric as well.

The last presentation was given by Bill Councill and focused on the changes that need to be introduced to traditional software processes in order to accommodate new approaches such as Component-Based Software Engineering (CBSE).

Standards are indispensable to facilitate the establishment of contracts between component producers and consumers, and should be associated with all phases of the software lifecycle. Third-party certification was also discussed as a method to establish conformance to standards. This is particularly important for small subcontractors that account for 99% of US businesses. It was stressed that organizational certification such as ISO9000 and CMM is organization/process related, not product/project specific and may say little about the meeting of specifications at the level of individual products.

Such a systematic approach to component-based software engineering can only be achieved by appropriate education of the parties involved, and by a clear assignment of their responsibilities. A new style of management is required: project and product managers should know —and should be able to perform— every phase of the CB lifecycle.

Open Discussion

Following these presentations, the session chair identified some drivers for metrics to be established for component-based software engineering. These include the possibility of evaluating the degree of trust that one may place on a component, for example by metrics associated with coverage achieved during testing with respect to a specific coverage criterion. Metrics would also make it possible to select among components with the same specifications.

3.7 Session 7: Modeling and Specification, Moderator: Clemens Szyperski

Software components seek to enable composition of software out of independently provided parts. The responsibility for the resulting compositions' meeting of requirements rests on multiple shoulders: each component provider asserts meeting some level of specification and the composer asserts that components were used according to their documented requirements and constraints. Proper modeling is at the heart of understanding requirements; proper specification is at the heart of sound assertions.

Papers presented in this session were:

- P. Kallio and E. Niemelä, "Documented Quality of COTS and OCM Components."
- M. Vieira, M. Dias, and D. Richardson, "Describing Dependencies in Component Access Points."
- D. Giannakopoulou and J. Penix, "Component Verification and Certification in NASA Missions."

This session focused on several modeling and specification aspects germane to software components. The session's first half covered three presentations ranging over a broad spectrum of topics; followed by a second half of discussion.

Päivi Kallio and Eila Niemelä provide a general template for documenting software components that considers both the buyer's and the provider's view. The template remains at an informal level, but encourages a certain degree of completeness of information by providing a checklist of points to consider. Information items covered include a diverse range from a component's history to performance characteristics.

In the discussion a number of open questions were raised by the audience and the presenter providing achors for further research exploration for this work in progress: relation to existing standards for component models and their documentation, domain specific requirements for documentation, and the need for separating documentation from the user and the reuser perspectives, given that reusers are typically developers.

Marlon Vieira, Marcio Dias, and Debra Richardson discuss issues related to component dependencies and introduce an approach to describe what can happen (in term of actions/dependencies) after a particular component's access points (services) are called. Their approach rests on formalizing certain dependency forms in specifications of a component's access points. Using this information about the diverse components' access points, they propose to construct dependence graphs, showing components in the nodes and actions in the edges.

These are then decorated by assertions. The authors identified as future steps for their ongoing research: development of taxonomy of dependencies, further extension of their proposed dependency definition language and mining and correlating existing component technologies with regard to elements of dependency elicitation.

The subsequent discussion recognized the similarity of some dependency definitions with assertions in Meyer's design-by-contracts approach and the need to limit assertional elements of the dependency definitions as the general problem of specification reengineering and transformations of incomplete specifications are notoriously hard.

Dimitra Giannakopoulou and John Penix discuss applications for NASA missions that combine ambitious scientific goals with requirements for high reliability. As a result, verification technologies are therefore taken to and pushed beyond their current limits. Also, to meet tight deadlines, reuse and adaptation of software architectures and components must be incorporated in software development within and across mis sions. While still at an early stage of their research, they already observe the importance of modularity, an inherent property of truly component-oriented architecture, for purposes of verification.

At the end of the presentation the presenters raised a number of open questions in particular relating to the level of detail, specification models, and decomposition suitable for hybrid military systems combining database elements with autonomous agents, command and control, and so forth. How for example should we certify a rover control system including adaptive technology such as neural networks.

3.8 **Workshop Summary Session,** Session moderator Heinz Schmidt

The final session of the workshop began with presentation of session summaries of each of the working sessions. This was followed by energetic discussion of where to go next. There was strong support for the creation of a model problem to be used as a research focus by the different communities represented at the workshop (software architectures, component technologies, formal methods, certification). The problem should allow each to demonstrate the strengths and weaknesses of their approaches on the model problem, and identify open research issues.

While time did not allow for the actual definition of the model problem, six criteria for the problem were identified.

- Amenable to custom as well as existing components
- Include extra-functional attributes
- Rich enough to demonstrate architectural prediction, not just component but emergent properties
- Allow precision and approximation
- Dynamic extensibility and evolvability
- Hypothesis should be defined in a way that can be validated.

Several model problems were proposed for consideration. George Heineman set up and continues to manage a discussion forum on his website at WPI. It was agreed that a 5th CBSE should be held at ICSE in 2002 as well as a mid-year workshop for the purpose of defining the model problem.

4 Future Plans

Several workshop participants continued to participate in the forum to the date of this writing and the midyear workshop will be held on October 19, 2001 at the Software Engineering Institute at Carnegie Mellon University in Pittsburgh, Pennsylvania. The problem presented by Otto Preiss “Towards a Composition Model Problem Based on IEC61850” is being strongly considered for use as the model problem because it is a well-defined problem, with requirements involving several quality attributes that need to be predicted on assemblies. Selection of the model problem will be finalized during the first session of the October workshop.

The workshop has been organized in order to support our continued collaboration and development of the model problem chosen after CBSE4. The workshop will consist of group discussion of issues related to predictable assembly and breakout groups to work on specific research areas that can be explored in order to contribute to a solution to the overarching problem.

The proceedings of the workshop are available on the web[5][6] and will be made available in hard copy as technical reports from the Software Engineering Institute and Monash University. 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 planned for early 2002.

5 Acknowledgments

Many people have contributed to this report. Section 3.1 was written based on material and presentation of **Jeff Voas**, **David Garlan**, and **Clemens Szyperski**. **George Heineman** and **Betty Cheng** submitted the summaries used for section 3.2**.** Section 3.4 is an extract of **Murali Sitaraman’s** submission**. Dave Wile** contributed to section 3.5**.** The section 3.6 is an extract of a submission of **Dimitra Giannakopoulou and Clemens Szyperski**. Finally, the summary report from **Clemens Szyperski** was used in section 3.7.

6 References

[1] Alan Brown, Kurt Wallnau, “The Current State of Component-Based”, Software Engineering (CBSE) IEEE Software, September 1998, pg. 37-47.

[2] http://www.sei.cmu.edu/cbs/icse98/index.html

[3] http://www.sei.cmu.edu/cbs/icse99/index.html

[4] http://www.sei.cmu.edu/cbs/cbse2000/index.html

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

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