Architectural Verification of Black-Box Component-Based Systems

Antonia Bertolino[1], Henry Muccini[2], and Andrea Polini[1]

[1] Istituto di Scienza e Tecnologie della Informazione "Alessandro Faedo"
Consiglio Nazionale delle Ricerche
via Moruzzi, 1 – 56124 Pisa, Italy
{antonia.bertolino, andrea.polini}@isti.cnr.it
[2] Dipartimento di Informatica,
University of L'Aquila
Via Vetoio, 1 - L'Aquila, Italy
muccini@di.univaq.it

Abstract. We introduce an original approach, which combines monitoring and model checking techniques into a comprehensive methodology for the architectural verification of Component-based systems. The approach works by first capturing the traces of execution via the instrumented middleware; then, the observed traces are reverse engineered into Message Sequence Charts, which are then checked for compliance to the Component-based Software Architecture, using a model checker. The methodology has been conceived for being applied indifferently for validating the system in house before deployment and for continuous validation in the field following evolution. A case study for the first case is here illustrated.

1 Introduction

Two antithetical approaches which emerge today for the verification of large complex distributed systems are *model-based* and *monitoring*. These two approaches are generally used in different stages of the software life cycle, and serve different purposes. The former enforces the rigorous derivation of a set of test cases from the system model, and aims at validating before deployment that the implemented system behaviour actually conforms to the modeled one. The latter collects and analysis runtime data during system execution to identify failures and to evaluate critical quality and performance attributes in the field.

In this paper we describe an original approach we are working on, which draws from both model-based verification and monitoring concepts, and combines their respective strengths into a comprehensive methodology for verification in a continuum between in-house and in the field. In particular we describe here how the behaviour of a component assembly is validated against the corresponding Software Architecture.

A Software Architecture (SA) provides high-level abstractions for representing the structure, behavior, and key properties of complex software systems [12]. SA-driven development assigns to the SA specification a central role in the software

N. Guelfi and D. Buchs (Eds.): RISE 2006, LNCS 4401, pp. 98–113, 2007.

life cycle, both to the phase of design and integration, and to analysis and testing activities. Most methodologies for SA-based analysis and testing generally assume a model-driven approach, in which the SA specification constitutes the reference model and the system is subject to a thorough and accurate validation of the required architectural properties before being deployed.

Advances in SA have greatly contributed to the advent of the *Component-Based paradigm* of development. In fact, the SA specification provides the blueprint for developing systems by properly composing "pieces" of software against it. A Component-Based Software System (CBS) can be roughly considered as an assembly of reusable components, designed to meet the quality attributes identified during the architecting phase [9]. A component can be defined [20] as a unit of composition, with contractually specified interfaces. In a CB approach, a big challenge is posed by the scarce information that is generally available about the components. Various approaches for testing CBSs have been recently proposed spanning over a varying spectrum of assumptions made on the metadata accompanying the component, which can be merely in form of pre- and post-conditions, or even as detailed as state machines. However, such a paradigm needs also to assume there is a time for validation before deployment; with the fast and ceaseless increase of systems complexity and pervasiveness, and the consequent emergence of dynamic global SAs, such a model need to be revised.

In antithesis to a proactive approach to testing such as model driven, several proposal for monitoring, or passive testing, are today spreading. Referred to with differing terminology, such as "monitoring", "tracing", and similar, what this approach to verification foresees is to observe the system during execution and to profile the obtained traces with different purposes. Hence, while in model-based testing the system must be stimulated so to reproduce some predefined behaviour, in monitoring the actual behaviour is observed and a posteriori analysed to see whether this conforms to desired properties. This prevents the burden of reproducing preselected test sequences as in model driven testing; we adopt the model-based approach in that we derive some architectural properties from the specification that we want to verify, and we verify the collected traces against these properties. In particular, for the latter purpose, we apply model-checking techniques, by which we check that the CBS derived traces conforms with the expected event sequences in the CBSA model.

The goal of such an approach is to ensure that the "core" of the implemented system fulfills the SA expectations, as figuratively illustrated in Figure 1; so, for instance, the approach can help to verify that by adding a new plugin component to a given CBS, its overall behaviour does not deteriorate. In light of the evolutionary properties of modern CBS, the same approach is meant to be used both at development stage, and after deployment.

In the next two sections we provide an overview of the proposed approach, and of related work. Then, in Sections 4 and 5 we describe the monitoring and model-checking steps, and in Section 6 we discuss the application of the approach to a case study. Finally some conclusions of results reached so far are drawn in Section 7.