Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

Science of Computer Programming 61 (2006) 16–26

Science of Computer Programming

www.elsevier.com/locate/scico

Detection of anomalies in software architecture with connectors

Michael E. Shin[*], Yan Xu, Fernando Paniagua, Jung Hoon An

Department of Computer Science, Texas Tech University, Lubbock, TX 79409-3104, United States

Received 4 March 2005; received in revised form 14 September 2005; accepted 7 November 2005
Available online 9 March 2006

Abstract

This paper describes an approach to detecting anomalies in a software architectural style that is structured with components and connectors between the components. Each component is designed with tasks (concurrent or active objects), connectors between tasks, and passive objects accessed by tasks. Anomalies in the software architecture are detected twofold by each Component Monitor, which supervises objects in a component, and by a System Monitor, which monitors message communications between components. The monitors encapsulate the specifications of objects being monitored, which are represented using statecharts. The execution of statecharts in the monitors depends on notification messages from connectors between tasks, passive objects accessed by tasks in a component, and connectors between components.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Anomaly detection; Software architecture; Connector; Component; Notification message

1. Introduction

The detection of anomalies in dependable systems is crucial to high quality service in those systems. Anomalies in a system can result from different types of causes such as design faults, system failure, malicious attacks, physical damage, low performance, or network congestion [8]. The important factors in the mechanism for anomaly detection are (1) how fast the mechanism detects anomalies and (2) how accurately it locates them in a system. Thus very dependable systems are in need of anomaly detection mechanisms that are capable of providing both high speed detection and accurate location of anomalous objects.

Several approaches to anomaly detection for dependable systems have been suggested in [3,8–10,7,4], which may provide partial solutions from the perspective of quality factors of the mechanisms for anomaly detection—speed and accuracy. Time testing, referred to as heartbeating [3,9,10], can be used to check if a component or system is anomalous, but it may fail to locate where an anomaly is in a component or system. The detection mechanism depending on exceptions [7] may not handle unanticipated, state-dependent anomalies.

This paper describes an approach to detecting anomalies in a software architectural style structured with components and connectors, which can meet quality factors of detection mechanisms such as speed and accuracy of anomaly detection. This begins by describing a software architecture style in Section 2. Section 3 describes the

[*] Corresponding author. Tel.: +1 806 742 3527.
E-mail addresses: Michael.Shin@ttu.edu (M.E. Shin), yan.xu@ttu.edu (Y. Xu), fernando.paniagua@ttu.edu (F. Paniagua), jh.an@ttu.edu (J.H. An).

0167-6423/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2005.11.002

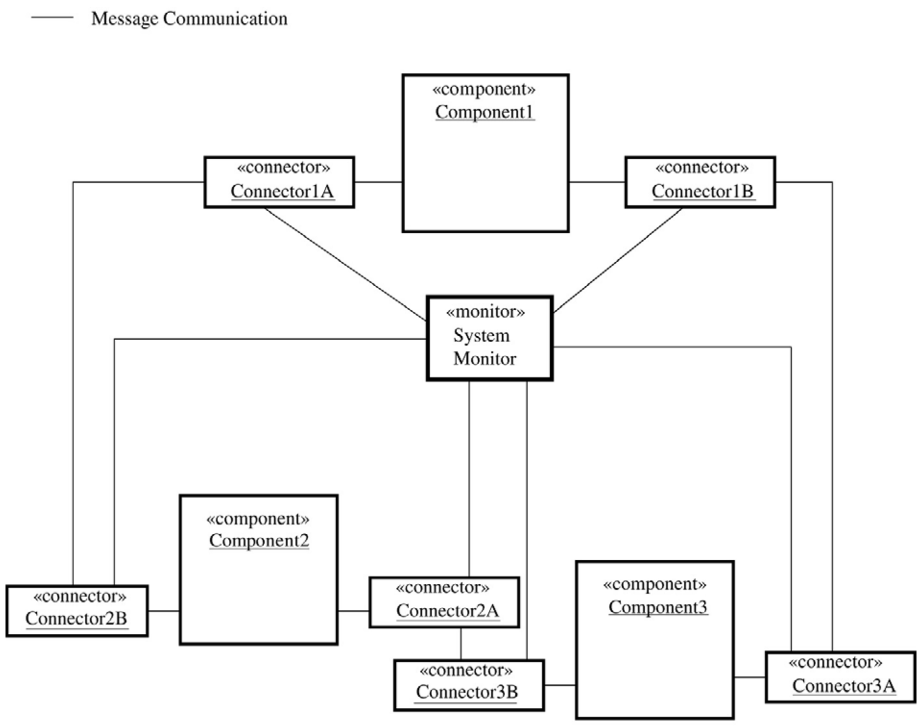

Fig. 1. Overview of software architecture with connectors.

overview of the approach suggested in this paper. Sections 4 and 5, respectively, describe the detection of anomalies within a component and in interactions between components of the software architecture. Section 6 describes the evaluation of our approach. Section 7 describes related work. Finally, Section 8 concludes this paper.

2. Software architecture with connectors

A software architecture [2,12] can be modeled by means of components and their interactions via connectors between the components. A component provides functional services to others. Messages passing between components are delivered via connectors, which synchronize the message communication between components. Fig. 1 depicts an overview of the software architecture with connectors, which is structured with three components, Component1, Component2, Component3, and their connectors. For example, Component1 has Connector1A and Connector1B for message communication with Component2 and Component3.

Each component in the software architecture with connectors can be designed with active objects referred to as tasks, passive objects accessed by tasks (e.g., entity objects storing data), and connectors between tasks [5]. Fig. 2 depicts objects in a component. A task has its own thread of control, initiating actions that affect other active and passive objects [5]. Unlike a task, a passive object has no thread of control; thus it cannot initiate any active objects. But a passive object is invoked by tasks and can invoke other passive objects. Because a passive object does not have its own thread, it performs its operations using the thread of the task that invoked the object. Tasks in a component can communicate with each other through connectors. On behalf of a task, connectors send messages to and receive them from other tasks. Like a connector between components, a connector between tasks within a component encapsulates the synchronization mechanism for message communication.

3. Approach to anomaly detection

An anomaly in the software architecture with connectors can be hidden in components and interactions between the components. The software architecture with connectors is anomalous if components or interactions between the components in the architecture do not behave as specified. The specifications of components and interactions between components in the software architecture are described using statecharts, which model the dynamic aspects of components and interactions between components by means of states and transitions with events.