Beagle

Glossary and Acronyms

Annika Berger, Joshua Gleitze, Roman Langrehr,
Christoph Michelbach, Ansgar Spiegler, Michael Vogt

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

Abbreviations

API Application Programming Interface

CPU Central Processing Unit

CTA Common Trace API

GUI Graphical User Interface

HDD Hard Disk Drive

JRE Java Runtime Environment

PCM Palladio Component Model

QoS quality of service

SEFF service effect specification

SRS Software Requirements Specififcation, see [Berger et al., 2015]

Terms and Definitions

assembly context

the components in conjuntion with a component. Specifies which components provide the component's required interfaces.

Beagle

"**BE**haviour **A**nalysis using **G**enetic **L**earning and **E**volution". Approach for dynamic analysis of source code in order to find its behavioural attributes developed in [Krogmann, 2011]. This project aims to implement Beagle.

ByCounter

tool that instruments Java bytecode and executes it in order to count how often each method and Java Byte Code instruction is called. The resulting counts may serve as fine-grained, deployment-independent information about the measured code's resource demands [Kuperberg et al., 2008].

Common Trace API

an API developed by NovaTec GmbH for measuring the time, specific code sections need to be executed.

component

"a [software] unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to third-party composition." [Szyperski, 2002]
There is no equivalent of components in modern programming languages, in particular, a component usually consists of multiple Java classes. Components can be nested.

component developer

"[specifies] the functional and extra-functional properties of their components. They put the specification as well as the implementation in repositories, where software architects can retrieve them." [Koziolek et al., 2007]

In the Palladio Component Model (PCM), component developers create service effect specifications to define components' behavioural properties and store modelling and implementation artefacts in repositories. [Reussner et al., 2011]

component-based software

a software constituted of components.

component-based software architecture

a software architecture utilising the concept of component-based software, therefore taking advantage of the reusability of its parts and preserving the same for newly created components.

deployment architecture

describes in which hardware node each component of a system is deployed.

deployment context

in which environment a component runs. Specifies which resources are available to the component. Includes information like the performance of the hardware the component runs on but also information about the software resources like the virtual machine or thread pools.

internal action

sequence of commands a component executes without leaving its scope (e.g. without calling other components). Part of a component's service effect specification (SEFF).

Java Runtime Environment

a software set containing a Java Virtual Machine, a browser plugin, the Java standard libraries, and a configuration tool. The Java Virtual Machine it contains is needed to run Java applications or applets.

Kieker

> "a Java-based application performance monitoring and dynamic software analysis framework." [van Hoorn et al., 2012]

A measurement software Beagle aims to support.

layer

describes conceptual separation in software.

logical architecture

describes a system's logical partitioning into layers, subsystems, and packages, and their communication with each other.

measurement software

software capable of measuring the time, given source code needs to execute some task. The software's results are usually returned in a time unit like nanoseconds. Beagles interacts with such software through the Common Trace API (CTA) and uses it to find resource demands.

Palladio

an approach to the predict quality of service (QoS) properties of component-based software architectures with a special focus on performance properties.

Palladio Component Model

a domain-specific modelling language (DSL) used by Palladio.

> It is designed to enable early performance predictions for software architectures and is aligned with a component-based software development process. [Kounev, 2009]

PCM source code decorator

realises links from the source code to the elements in the PCM and the other way round. [Krogmann, 2011]

PCM Stochastic Expression Language

expression language used by the PCM to define random variables. These variables can for example be used to specify glsplresource demand. Random variables can be defined using basic mathematic operations, common stochastic distributions and interface parameters [Reussner et al., 2011].

quality of service

a software's extra-functional attributes, like performance, reliability, maintainability or security.

resource demand

how much of a certain resource—like Central Processing Unit (CPU), Network or hard disk drive—a component needs to offer a certain functionality. In the PCM, resource demands are part of the SEFF. They are ideally specified platform independently, e.g. by specifying required CPU cycles, megabytes to be read, etc.

If such information is not available, resource demands can be expressed platform dependent, e.g. in nanoseconds. In this case, a certain degree of portability can still be achieved if information about the used platforms' speed relative to each other is available.

SEFF condition

conditions (like Java's `if`, `if-else` and `switch-case` statements) which affect the calls a component makes to other components. Such conditions are—contrary to conditions that stay within an internal action—modelled in the component's SEFF.

SEFF loop

loops (like Java's `for`, `while` and `do-while` statement) which affect the calls a component makes to other components. Such loops are—contrary to loops that stay within an internal action—modelled in the component's SEFF.

service effect specification

description of a component's behaviour in the PCM. SEFFs contain information about the component's calls to other components as well as its resource demands. This information is used to derive the component's performance for simulation and prediction.

software architect

developer role in the component-based software development process. Leads the development process by designing the software's architecture from existing or planned components and interfaces. Usually delegates the specification of required components to component developers. Uses architectural styles and patterns, analyses architectural specifications, and makes design decisions. In the PCM, software architects create the assembly model, specifying how existing components are composed. [Reussner et al., 2011]

software architecture

the high-level structure and design of a software system as well as the discipline of creating and documenting these.

SoMoX

"**So**ftware **Mo**del e**X**tractor", a Palladio plugin for static code analysis to re-engineer a software's architecture from its source code developed in [Krogmann, 2011]. Constructs a PCM instance including the reconstructed components and their SEFF.

subsystem

a self-contained system within a larger system.

system

the useful whole created from diverse parts. A system (usually) reflects the organizational structure that built it. (Conway's law) [Conway, 1968]

system deployer

developer role in the component-based software development process. Specifies the resource environment and allocates components to resources. Resources can both be hardware resources (CPU, hard disk, network connection) and software resources (thread pool, database connection). In the PCM, system deployers create the resource environment specification, modelling the resource environment and component allocations. [Reussner et al., 2011]

tier

consists of a server or group of servers. Tiers are physically separated from each other.

usage context

how a component is used. Includes the number, frequency and distribution of calls made the the component's services.

Bibliography

[Berger et al., 2015] Berger, A., Gleitze, J., Langrehr, R., Michelbach, C., Spiegler, A., and Vogt, M. (2015). Beagle—software requirements specification. Technical report, Karlsruhe Intitute of Technology Department of Informatics Institute for Program Structures and Data Organization (IPD).

[Conway, 1968] Conway, M. E. (1968). How do committees invent? *Datamation*.

[Kounev, 2009] Kounev, S. (2009). *Automated extraction of palladio component models from running enterprise Java applications*. PhD thesis, University of Wuerzburg.

[Koziolek et al., 2007] Koziolek, H., Happe, J., Becker, S., and Reussner, R. (2007). *Palladio Paper*. PhD thesis.

[Krogmann, 2011] Krogmann, K. (2011). *Reconstruction of Software Component Architectures and Behaviour Models using Static and Dynamic Analysis*. PhD thesis, Karlsruhe Institute of Technology.

[Kuperberg et al., 2008] Kuperberg, M., Krogmann, M., and Reussner, R. (2008). ByCounter: Portable Runtime Counting of Bytecode Instructions and Method Invocations. In *Proceedings of the 3rd International Workshop on Bytecode Semantics, Verification, Analysis and Transformation , Budapest, Hungary, 5th April 2008 (ETAPS 2008, 11th European Joint Conferences on Theory and Practice of Software)*.

[Reussner et al., 2011] Reussner, R., Becker, S., Burger, E., Happe, J., Hauck, M., Koziolek, A., Koziolek, H., Krogmann, K., and Kuperberg, M. (2011). The palladio component model. Technical report, Karlsruhe Intitute of Technology Department of Informatics Institute for Program Structures and Data Organization (IPD).

[Szyperski, 2002] Szyperski, C. (2002). *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition.

[van Hoorn et al., 2012] van Hoorn, A., Waller, J., and Hasselbring, W. (2012). Kieker: A framework for application performance monitoring and dynamic software analysis. In *Proceedings of the 3rd joint ACM/SPEC International Conference on Performance Engineering (ICPE 2012)*, pages 247–248. ACM.