Available online at www.sciencedirect.com
SCIENCE DIRECT
ELSEVIER
Electronic Notes in Theoretical Computer Science 114 (2005) 47–64
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Towards Composition Management for Component-based Peer-to-Peer Architectures

Sascha Alda [1] Armin B. Cremers [1]

Institute for Applied Computer Science
University of Bonn
Bonn, Germany

Abstract

Recent peer-to-peer architectures do not fulfill the idea of a service-oriented architecture to allow the flexible composition of services towards concrete applications. This can be justified by the absence of flexible notations for the composition of services that incorporate the dynamic nature exposed by peer-to-peer architectures. In this work, the peer-to-peer architecture DeEvolve is presented that provides novel ways for the composition of services including the handling of exceptions such as the failure of peers. The intention of this approach is to facilitate even less-skilled end-users to compose and to maintain service-oriented applications.

Keywords: Component technology, composition languages, peer-to-peer architectures, exception handling.

1 Introduction

Software architectures represent software systems as a coherent set of high-level computational elements such as components, objects, or services together with their interactions and dependencies. The merit of an architecture-based development is to drift away from a low-level, code-based development towards a more flexible development focusing on the *composition* of self-contained building blocks. The structure of software architectures is often specified in a declarative manner by an external, formal notation. Prominent approaches for these notations are architecture description languages (ADLs) or workflow

[1] Email: {alda, abc}@cs.uni-bonn.de

1571-0661/$ – see front matter
doi:10.1016/j.entcs.2004.02.067

languages. Each approach comes along with appropriate assembly or adaptation tools in order to support the process of realizing software architectures even for less skilled end-users.

The notion of a software architecture makes at first no proposition concerning the actual organization, distribution, or availability of the constituting building blocks. Therefore, designers usually revert to so-called *architectural styles* to design software architectures on top of appreciated and established architectural organizations [5]. Well-known architecture styles for instance are the client-server, layered, or pipe and filters style. The selection of a style depends strongly on the intended use of the architecture.

In this work, the peer-to-peer paradigm [10] [2] is examined as another architectural style. Following this style, a peer-to-peer architecture constitutes a distributed architecture that consists of equal clients or so-called *peers*. Peers are capable not only of consuming, but also of providing computer resources that are encapsulated by *peer services*. In contrast to other architectures, peer-to-peer architectures assume an unstable, dynamic topology as an important constraint. Peers are solely responsible to affiliate to a peer-to-peer network. This degree of freedom permitted for peers could lead to unanticipated behavior or *exceptions* within the whole peer-to-peer architecture.

Though recent peer-to-peer architectures do already provide for various options for the interaction between peers, they exhibit two major drawbacks. First, neither architecture allows for the composition of peers (or their offered services) by means of appropriate notations. Moreover, current architectures are hardly resistant against unanticipated exceptions such as the failure of single peers, which is due to the absence of sophisticated models for exception detection and resolution. On the other hand, existing composition languages are not sufficiently flexible to describe compositions of services that reside within an unstable environment as typical for peer-to-peer architectures. There is also no explicit exception handling to describe alternative configurations in an exceptional case.

The major contribution of this work is a novel approach for the composition of peer services within a peer-to-peer architecture to overcome the drawbacks of existing architectures and notations. A notation called PeerCAT is presented for the composition of various peer services in a declarative manner. PeerCAT also allows to define *exception handlers* itemizing resolution plans in the case of exceptions caused for instance by the failure of peers. The structure of peer services is modelled by the composition of components. A component model does thereby prescribe the valid remote and local interaction primitives between services in a unified way. Both the component model and PeerCAT constitute the foundation for the component-based runtime environment DeE-