An Extension of a CCM Environment with an Adaptive Planning Mechanism

Paweł Antoniewski, Łukasz Cygan, Jacek Cała, Krzysztof Zieliński
Department of Computer Science
AGH – University of Science and Technology
al. Mickiewicza 30, 30-059 Cracow, Poland

***Abstract*—The presented paper describes a promising approach of adaptive deployment in CORBA Component Model (CCM). Adaptation, as an essential concept of contemporary distributed systems, is used to improve planning of component deployment. There is described a concept and implementation of mechanisms which, using monitoring infrastructure and simple policy-based engine, allows improving execution efficiency. The presented tool, named CCMAD, is intended for use with OpenCCM platform, an open source Java implementation of CCM model. It is, however, expected that adapting CCMAD for any other CCM platform would not be troublesome.**

I. INTRODUCTION

Nowadays, adaptation of distributed applications to execution environment is very important issue in more and more complex computer systems sharing resources between many concurrent users. Efficient process of resources management in large computer systems is still very challenging task which draws attention of research community. This manifests in new research areas related to Next Generation Grid Initiative [12], Autonomic Computing proposed by IBM [10], Utility Computing started by SUN [14] and many others. All these initiatives recognize adaptation of applications as one of the most demanded features, which may influence almost every phase of application life-cycle.

This work discusses how deployment of component-based applications may benefit from adaptability. In this paper the broad vision of adaptability is limited to compositional adaptation [7] which determines how to modify structure of the software in response to changes in its execution environment. Despite that there are several other approaches providing application adaptation [2][3], component-based architectures are particularly convenient to introduce compositional adaptation. This is mainly due to distinct borders which define a component as well as inherent support for late binding [5]. In consequence, component-based software engineering allows effectively applying techniques such as redeployment and migration to improve application execution [4][6].

The proposed extension is targeted to CORBA Component Model (CCM) [8]. The CCM computational model represents one of the most advanced distributed component environments what fully justifies its selection as a subject of this study.

The defined CCM deployment process assumes that an application in form of assembled components is started over a target execution domain composed of connected computational nodes. The whole process consist of a few steps of which the planning is the most troublesome and complex. Deployment planning is a process of allocation of application component to the nodes and typically it relies on a static description of target domain resources.

The paper describes an extension of a CCM environment with mechanisms supporting adaptive application planning that take into account not only resource specification but also their current load and availability.

The text is structured as follows. In Sect. II there is briefly presented deployment of CCM applications. This creates background for the proposed concept of adaptive deployment mechanism presented in Sect. III. The constructed software tool, named CCMAD, supporting practical implementation of adaptive deployment process is described in more details in Sect. IV. The case study which compares efficiency of a testing application performance run by standard CCM deployment procedure with the same application deployed by CCMAD is presented in Sect. V. The paper is ended with conclusions.

II. PROCESS OF DEPLOYMENT OF CCM APPLICATIONS

The aim of deployment is to distribute, install, configure and connect software in a way which enables an application to execute in a target environment. OMG in *Deployment and Configuration of Component-based Distributed Applications specification* (D&C) [9] defines a course of this process for component-based applications which is further specialized in CORBA Component Model. Previously, however, CCM defined much more simplified approach to deployment of components.

According to D&C specification, deployment procedure consists of five steps: (1) installation in a software repository,

T. Sobh (ed.), Innovations and Advanced Techniques in Computer and Information Sciences and Engineering, 379–383.

(2) configuration of default package properties, (3) planning how and where the software will run, (4) preparation the target environment to execute the application, and finally, (5) launching the software in the target which includes instantiation, configuration and interconnection of the application components.

This paper turns attention to the planning phase, as the most complex part of the deployment process. Successful planning results in a *DeploymentPlan* which is a mapping of each component of an application to a selected node available in a target execution domain. In this paper we show that effectiveness of the application execution might strongly depend on the way the components are arranged in the domain.

The main issue of planning the deployment is to match component requirements to suitable resource satisfier properties of the execution environment. Using D&C specification, developers have considerable degree of freedom in expressing component requirements and target domain properties. Both, *Requirement*s and resource properties in form of *RequirementSatisfier*s are composed of a list of name-value pairs and a type as a string of characters. The satisfiers have also additional information about their kind (e.g. capacity, quantity) as well as whether they are dynamic or static attributes (listing below shows requirement satisfier of a sample resource). This very general approach allows defining potentially any kind of a resource or requirement but also raises the issue of a common language to express them conveniently such that they are comparable and might be matched against each other.

Listing 1. An example of requirement satisfier of a CPU resource

```
<resource>
  <name>HostA CPU</name>
  <resourceType>cpu</resourceType>

  <property>
    <name>number</name>
    <kind>Quantity</kind>
    <value>
      <type><kind>tk_ulong</kind></type>
      <value><ulong>1</ulong></value>
    </value>
  </property>
  <property>...</property>
  ...
</resource>
```

Apart from the new approach of deployment defined in D&C, previously, CORBA Component Model introduced much more simplified way of describing application components. There were no means to define requirements of a component, there were also no means to describe a target environment. As a consequence, previous CCM specification did not undertake planning phase at all providing no means to describe a deployment plan.

Nevertheless, in order to run a component application in distributed environments, platforms implemented according to the older approach were forced to supply solution to this deficiency. This article is based on OpenCCM platform [13], to the best of our knowledge, the most advanced freely available Java implementation of CORBA Component Model. OpenCCM, based on the previous deployment procedure, extends the specification by defining an additional element of Component Assembly Descriptor (CAD). Therefore, a developer is able to assign an eligible destination node to a *homeplacement*, *hostcollocation*, or *processcollocation*.

A home placement element is a standard measure to describe a group of component instances managed by a single common component home, whereas process and host collocations allow defining bonds between two or more home placements. This binding between the placements enforces a home to be run in the same process or on the same machine respectively. Listing below presents an excerpt from a sample CAD file with the OpenCCM extension – a *destination* element.

Given such supplemented assembly descriptor, which now might be perceived as a simple deployment plan, OpenCCM deployment tools are able to distribute, install, configure and run application components on desirable component servers running on particular nodes.

Listing 2. The destination element – the extension to original Component Assembly Descriptor file

```
<componentassembly id="myApplication">
...
<partitioning>
  <homeplacement cardinality="1"
                 id="ConsumerHome">
    <componentinstantiation id="aComponent">
      ...
    </componentinstantiation>
    ...
    <destination>
      <installation type="component installation">
        <findby>
          <namingservice
                        name="ComponentServer1" />
        </findby>
      </installation>
    </destination>
  </homeplacement>
  ...
</partitioning>
<connection>
  ...
</connection>
```

III. Concept of Adaptive Planning Mechanisms for CCM

The analysis of the deployment procedure presented in the previous section leads to the conclusion that two most important issues which a planner should perform are: matching the requirements of application components to the resources in a target execution environment and then