

## Dynamic Architectural Adaptation Using Ontologies

José Luis Pastrana<sup>1</sup>, Ernesto Pimentel<sup>1</sup> and Miguel Katrib<sup>2</sup>

<sup>1</sup> Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga  
ETSI Informática, Campus Teatinos. 29071 Malaga, Spain  
{pastrana, ernesto}@lcc.uma.es

<sup>2</sup> Departamento de Computación I, Universidad de la Habana,  
Facultad de Matemática y Computación. San Lázaro y L, Vedado. 10400 C.Habana, Cuba  
mkm@matcom.uh.cu

**Abstract.** Software Adaptation promotes the use of specific computational entities called adaptors that guarantee software components will interact in the right way, not only at the signature level, but also at the behavioral, semantic and service levels. Adaptation techniques have proceeded by computing adaptors for closed systems made up of a fixed set of components. This is not satisfactory when the systems may evolve, with components entering or leaving it at any time. To enable adaptation on such systems, we propose one implementation of adaptors under .NET platform and we show how using an ontology one adaptor can be improved in order to achieve a runtime adaptation at service level in a developed system when a component is missing or replaced by other. This capability will increase the Quality of Service (QoS), making the system be Fault Tolerant.

**Keywords:** Ontology, Dynamic Architectural Adaptation, Component Composition, .NET.

### 1 Introduction

Component Based Software Engineering (CBSE) receives a great interest from the software engineering community. The aim is to create a collection of reusable components that can be used for component-based application development. Application development then becomes the selection, adaptation and composition of components rather than implementing the application from scratch.

The development of such a market of software components (similar to the market of hardware components and electronic devices) has always been one of the myths of Software Engineering, but it has never become a reality. The reason is that unlike what happens with hardware components, software is never reused “as it is”, but a certain degree of adaptation is always required. In such a situation, an adaptor is necessary in order to overcome compositional mismatches and to make components which otherwise cannot be plugged together composable.

We can see that “glue” techniques are required to adapt components that do not fit the compositional requirements of a system. However, it is not always possible to interconnect components in a desired way: the plugs of two (or more) components may not be plug-compatible. Consider the known problem of travelers which are unable to plug the razor they use at home into the plugs of various other countries. In such a situation, adaptors are needed to bridge the different interfaces. These kinds of problems are often referred to as architectural mismatch [1].

Successful composition of components does not necessarily imply successful interoperation: Interoperability is the ability of software components to communicate and cooperate with each

other [2]. Reconsider the problem with the razor: in some countries, different voltages are used and prohibit compatibility even with an adaptor: composition is possible, but interoperability is not. These situations require a transformer to transform the incompatible voltage. We will denote these kinds of problems as interaction or interoperability mismatch.

Both architectural and interoperability mismatch are part of a problem domain which can be denoted as compositional mismatch [3]. A compositional mismatch occurs whenever it is impossible to successfully interconnect components with existing connectors. Based on the discussion and the observations of the example, we define one Adaptor (as well called glue code) as follow: One Adaptor is the part of an application which overcomes compositional mismatches.

This proposal is based on the assumption of the components in the system are black boxes and conceptually distributed. So, the objective of this proposal is to solve the problem existing when the system is already developed and the adaptors needed are generated (automatically or ad hoc) and one (or more) component in the system is replaced dynamically and then an architectural mismatch occurs.

We show how architectural mismatch occurred when replacing a component can be solved adding an ontology of the domain of the problem to the adaptor. So, the adaptor can solve the architectural mismatch automatically, and transparent to the system, at runtime. We are presenting, as well, the implementation of this technique using C# under .NET platform.

This paper is organized as follows. In Section 2 we give a presentation of the ontologies and the semantic web. Next, Section 3 shows how ontologies can be used for architectural mismatch adaptation at runtime. Section 4 uses an example about rectangle triangles to illustrate the details about implementation. Section 5, is a discussion about related works, and, finally, Section 6 presents the conclusions and future work.

## **2 The Semantic Web**

The objective of the Semantic Web is to provide languages to express the content of Web pages and to make accessible to agents and computer programs the information that those pages contain. More precisely, the Semantic Web is based on a set of languages such as RDF and OWL that can be used to markup the content of Web pages. These languages have a well-defined semantics and a proof theory that allows agents to draw inferences over the statements of the language.

The second element of the Semantic Web is a set of ontologies, which provide a conceptual model to interpret the information provided. For example, an ontology of weather may contain concepts such as temperature, snow, cloudy, sunny and so on. It may also contain information on the relation between the different terms; for instance, it may say that cloudy and sunny are two types of weather conditions. The vision of the Semantic Web is the transformation of the Web into an Internet wide knowledge representation system, in which web pages provide information and ontologies provide the conceptual framework needed to interpret that information.

Semantic Web technology [4], [5] helps us in managing information in our business, personal space, or employment. The Semantic Web initiative aims to build a Web to be consumed by software, instead of humans as the traditional WWW does. Thanks to formal and semantic specification of the content, new applications such as product catalogues integration, advanced search and retrieval applications, knowledge inferences become possible. Even if the vision of the Semantic Web is still at its beginning the number of businesses that take benefit is rapidly increasing.

In the context of computer and information sciences, an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. An ontology specifies a vocabulary with which to make assertions, which may be inputs or outputs of knowledge agents (such as a software program).

Ontologies [6] can, therefore, be shared and reused among different applications. An ontology describes the subject matter using notions of concepts, instances, relations, attributes, and axioms. Concepts in the ontology are organized in taxonomies through which inheritance mechanisms can be applied.

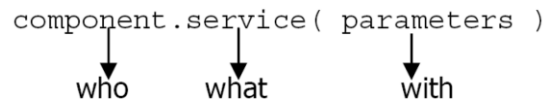
Ontology description languages such as RDF and OWL [7] are built on top of XML and add semantics to the model representation. Those languages are now standards of the W3C with a large support of applications and infrastructure for their storage, manipulation and overall management.

### 3 Using Semantic Web for Architectural Mismatch Adaptation

The web semantic technique used for the dynamic component adaptation of architectural mismatch is based on a formal specification of the semantic information, and, this specification is expressed using ontologies.

Ontologies have already been used in the context of translation. The idea of using ontologies to solve the adaptation problem is born from the fact that the only knowledge we have to make that adaptation is the component interface: service names, parameters, etc.

The grammar used in natural language is wide and complex; however, programming languages have a well defined grammar. In our particular case, we only have to keep in mind the invocation grammar which is as shown in **Fig. 1**.



**Fig. 1.** This figure shows the grammar used for the invocation of a service of a component

That allows us to adapt the call to a "not found" service by other one which is semantically equivalent. Adaptation works as follow: the adaptor catches the "not found service" exception and looks for an equivalent one in the current server using the unification algorithm of prolog machine between the request service and the knowledge set in the ontology and then invokes the new service.

There are many ontology languages but we have chosen the Ontology Web Language (OWL) because it is a W3C recommendation. OWL is a language for defining and instantiating Web ontologies. An OWL ontology may include descriptions of classes, along with their related properties and instances. OWL is designed for use by applications that need to process the content of information instead of just presenting information to humans. It facilitates higher machine interpretability because it is written in XML.

Looking at the features found in OWL, we find the `subclassof` relationship and the equality and inequality properties. The equality properties allow us to set which concepts are equivalent to other











