Towards Architectural Connectors for UML

Fernando Orejas[1] and Sonia Pérez[1,2]

[1] Departament LSI,
Universitat Politècnica de Catalunya,
Barcelona, Spain
[2] Instituto Superior Politécnico José Antonio Echevarría,
Havana, Cuba
{orejas,sperezl}@lsi.upc.es

Abstract. The notion of architectural connector was developed by Allen and Garland [1] as an important concept for the design of software architectures. In this paper, based on previous work introducing a generic approach for the definition of component-based concepts, we study how architectural connectors and components can be defined for class and sequence diagrams as a first step for applying this approach to full UML. In particular, the case of sequence diagrams is studied with some detail. A case-study of a lift system is used to illustrate these ideas.

1 Introduction

The development of component-based systems is nowadays an important area in software engineering. In this context, a lot of work has been dedicated to different issues related to approach, such as the development of methodologies and the implementation of middleware and other related tools. However, much less work has been dedicated to the modelling and specification phase of this kind of systems. In this sense, an approach that we consider very interesting is based on the use of *architectural connectors* [1, 8]. In this approach architectures are built in terms of two kinds of units: components and connectors. Components are not connected directly, but through connectors. Components offer some functionality and connectors describe policies of interaction of the connected components. Originally work, the language used for the specification and modelling of components and connectors was CSP [6]. The work using this approach was followed by Fiadeiro (e.g. see [5]), who generalized in some sense the approach by putting it into a categorical context in the framework of the coordination language COMMUNITY.

In [3], we developed a very generic approach for the definition of components whose aim was to allow the definition of component concepts associated to arbitrary formal or semiformal specification methods. The idea was that one could instantiate this generic approach to any arbitrary method, as long as one could prove that it satisfied certain properties. In particular, different instantiations were sketched in terms of Petri Nets, graph transformation systems or algebra transformation systems. We also studied how the approach in [3] could be used

H.-J. Kreowski et al. (Eds.): Formal Methods (Ehrig Festschrift), LNCS 3393, pp. 352–369, 2005.

to define generic architectural connectors. In particular, preliminary results, including an instantiation to Petri Nets, were presented in [4].

In the case of UML [7], a de facto standard for many industrial applications, the notion of component [2] has a very limited nature, being essentially a kind of syntactic package. In this paper, we present some basic ideas for the instantiation of our approach to the case of UML. In particular, we study how we can define architectural connectors and components in the case of class and sequence diagrams.

In particular, with very little detail in the case of class diagrams and more detail in the case of sequence diagrams, we study how the notions of embedding and transformation (which are key notions in our approach) can be defined. And we show that these notions satisfy the required properties. Especially, the so-called extension property.

The paper is organized as follows. In the next section we review our generic approach to components and architectural connectors. First we describe the approach presented in [3] and then, in the second subsection, we see how this approach can be adapted to define architectural connectors. The third section is the main one. First, we sketch the instantiation of our approach to the case of class diagrams and, then, we study sequence diagrams in some detail. In section 4, we present a small case study, a lift system, to show how these concepts can be applied in practice. Finally, in the last section, we draw some conclusions.

2 A Generic Framework for Architectural Components and Connectors

In this section we will describe the generic component framework introduced in [3], including some variations and extensions needed in this paper. More precisely, in the first subsection we will introduce our generic approach to architectural components and, in the second subsection, we will show how to represent architectural connectors in this framework.

2.1 Generic Components

Components are self-contained units, where some details are hidden to the external user. This is achieved by providing a clear separation between the interfaces and the body of the component. There are two kinds of interfaces: import and export interfaces. The export interfaces describe external views of the behavior of the component including the services that a component offers to the outside world. On the other hand, the import interfaces describe what the component assumes about the environment, including the services used inside the component that are assumed to be provided by other components. The interfaces are used to interconnect components. In particular, we can connect or compose two components by matching an import interface of one component with an export interface of the other component [9]. In [3] we assumed that each component had just one import and one export interface. On the contrary, in this paper we