Modelling Interoperability by CHAM: A Case Study

Paola Inverardi and Daniele Compare

Dip. di Mat. pura ed Appl., Università di L'Aquila
via Vetoio, Coppito, I-67010 L'Aquila, Italy.
(inverard@univaq.it), (compare@univaq.it)

Software architectures (SA) descriptions can be useful tool for the design, the analysis and the production of systems whose construction is based on the interoperability paradigm. In this paper we consider the SA of a system obtained by suitably assembling together existent software. The Cern Compressing Proxy (CP), has been presented in [4] and it is an interesting case study since it was obtained in two steps. The first version considered the interface mismatch between the components, which seemed to be the only problem to be solved in order to compose the existing pieces. Actually, when at work this solution exhibited a wrong dynamic behaviour since there were cases in which the whole system deadlocked. The second solution therefore removed this problem by analysing the dynamic behaviours of the components, thus suitably adapting the functionalities of the connecting new code. This example clearly shows the need of a precise description of the dynamic behaviour of the components that we claim should be retrieved from the architectural description. In the following, the descriptions of the two software architectures on which the CP was based on are given by using the Chemical Abstract Machine (CHAM) formalism [2] as introduced in [5]. A CHAM description of a software architecture consists of a syntactic description of the static components of the architecture (the molecule) and of a set of reaction rules which describe how the system dinamically evolves through reaction steps. In these descriptions both the origin of deadlocks in the first architecture and the ways to prevent them in the second architecture can be easily outlined and a certain amount of formal reasoning can be carried on. A CHAM is specified by defining molecules $m, m', \ldots$ and solutions $S, S', \ldots$ of molecules. Molecules are the basic elements of a CHAM, while solutions are multisets of molecules interpreted as defining the states of a CHAM. A CHAM specification contains transformation rules $T, T', \ldots$ that define a transformation relation $S \rightarrow S'$ dictating the way solutions can evolve (i.e., states can change) in the CHAM. At any given point, a CHAM can apply as many rules as possible to a solution, provided that no molecule is involved in more than one rule. Thus it is possible to model parallel behaviors by performing parallel transformations. When more than one rule can apply to the same molecule or set of molecules the CHAM makes a nondeterministic choice as to which transformation to perform.

The Compressing Proxy system is an example of interoperability obtained by suitably assembling together special http proxy servers, composed of many filters which are connected together via a function-call-based stream interface and gzip, which behaves as a traditional, standard input/output interface, Unix filter, i.e. it accepts a stream of input and produces a stream of compressed/decompressed output. The purpose of this system is to improve the performance of a Unix WWW browser connected to the Web by a slow link. This
The Compressing Proxy

This system when executed will sometimes deadlock. This happens because the entire document is sent to gzip before any attempt to read from it is done. If gzip tries to output before the adaptor has finished to send data out, gzip will block waiting for the adaptor to read. When then the adaptor tries to send more output to gzip, httpd will block waiting for gzip, thus causing a deadlock. This problem can be avoided by changing the communication modalities of the adaptor towards gzip, i.e. by using of non-blocking read and writes.

In the first architecture the adaptor uses blocking writes when sending data to gzip. We must define an algebra of molecules i.e. a syntax by which molecules can be built. We start with a set of constants \( P \) representing the processing elements and an infix operator "\( \circ \)" that we use to express the status of a processing element. The connecting elements for the blocking architecture are given by a the set \( C \) consisting of two operations, \( i \) and \( o \), that act on the elements of the set \( N \), that give the topology of the sistem, i.e. the communicating channels that connect components. The set \( E \), introduces the costants used in the communication between gzip and the adaptor. The syntax \( \Sigma_b \) of molecules \( M \) is