Software Architecture of the EPOCA Integrated Environment

Susanna Donatelli¹, Giuliana Franceschinis¹, Nicola Mazzocca² and Stefano Russo²

¹ Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, 10149 Torino - Italy
² Dipartimento di Informatica e Sistemistica, Università di Napoli
Via Claudio 21, 80125 Napoli - Italy

Abstract. We describe the software architecture of EPOCA (Environment for analysis and Performance evaluation Of Concurrent Applications), a tool for the analysis of concurrent programs. The analysis is based on a formal model of the application. The class of models chosen is that of stochastic Petri nets (in particular we adopt Generalized Stochastic Petri Nets - GSPN [1]): starting from a concurrent program written in DISC (DiStributed C), an extension of C to include concurrent constructs of the CSP type [12], a GSPN model is automatically generated, and GSPN analysis tools can then be applied. EPOCA is built as an integration of the DISC environment (a graphical interface based environment that provides compiling, monitoring and profiling facilities for DISC programs) and GreatSPN [6] (a graphical interface based environment for the definition and the analysis of GSPN).

1 Introduction

The EPOCA (Environment for analysis and Performance evaluation Of Concurrent Applications) tool presented in this paper results from the integration of a distributed program development environment based on C, called DISC [11] (DIStributed C), with a modeling and performance analysis tool, called GreatSPN [6], which is based on a class of stochastic Petri nets called GSPN [1].

EPOCA supports performance oriented distributed program development, by offering the possibility of studying the behavior of an application during different stages of its implementation cycle. The analysis performed is of static type [14] although dynamic analysis is also possible in EPOCA exploiting the monitoring tool provided by DISC. Static analysis can be used since the early stages of the program development because it doesn’t require a detailed implementation to be available; this analysis technique employs a probabilistic approach to data

* This work has been supported in part by the CNR project “Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo, Sottoprogetto 3: Architetture Parallele” and by the Italian Ministry of University and Scientific and Technological Research within the “40%” Project.
dependency, hence it gives average results. Monitoring on the other hand, is 
used at an advanced implementation stage, is very data dependent, and hence 
may give more precise and detailed results than static analysis with respect to 
specific input data. It is not well-suited to obtain average results because this 
requires the generation of several program execution traces. Interaction between 
static and dynamic analysis tools is possible: traces produced by the monitor 
might be used to derive timing and probabilistic information to be introduced 
in the abstract model; on the other hand, the model could be used as a basis 
for visualizing monitor traces at a fairly high level of abstraction, for example 
to visualize a trace that led to a deadlock.

In order for static analysis to produce reliable results, the formal model used 
for this purpose must be a consistent representation of the program behavior. 
Our tool guarantees the desired consistency by automatically generating the 
models through translation of the process activation and interaction scheme 
described in the program code. The automatic translation approach has several 
advantages: it does not require the programmer to be an expert in Petri net 
modeling, it avoids the boring and error-prone manual model input phase, and 
finally, it avoids the risks of using a program model that does not represent the 
actual program behavior but rather the behavior that the programmer thinks 
the program should have.

The final program can also be studied by monitoring executions through the 
DISC monitor.

The program model is generated in two steps: the first step is embedded into 
the DISC compiler and consists of the translation of each process in isolation 
using well defined DISC statements translation rules to produce GSPN models 
of each process scheme. The second step is performed by the so called “Petri net 
linker" and generates the complete program model by properly instantiating and 
linking the several process models produced in the first step. The output of the 
PN linker is again a GSPN description file that can be read by the GreatSPN 
tool.

From GreatSPN it is possible to run several analysis algorithms and apply 
the available facilities to validate the model and study its performance. The 
analysis results may give hints on how to modify the program (in case some 
unexpected or inefficient behavior of the program is found) or they may provide 
useful information for a subsequent (semi)automatic mapping phase (for example 
as suggested in[8, 5]). The monitor provided by EPOCA, inherited from DISC, 
consists of a module that captures the events naturally handled by the run-time 
environment. Therefore measurement data are obtained without need of program 
instrumentation, thus keeping the degree of intrusiveness at a minimum level.

In this paper the features of EPOCA are presented and the details of the 
programs-to-GSPN translation module, that acts as a bridge between the DISC 
and GreatSPN tools, are explained.

The paper is organized as follows: Section 2 presents the characteristics of 
the two systems that are more relevant to the final tool, moreover it explains 
the structure of the tool. Special emphasis is placed in Section 3 on the software