RATIONAL DESIGN OF DISTRIBUTED APPLICATIONS

Thierry Cattel
Digital Equipement, Centre Technique Europe Sarl
Chemin du Levant, 01210 Ferney-Voltaire
Laboratoire d'Informatique, URA CNRS 822
Université de Franche-Comté
25030 Besançon cedex
France
Phone : +33 81.66.64.58  Fax : +33 81.66.61.77
E-mail : cattel@emc2.enet.dec.com

Abstract

Although numerous distributed programming facilities have been developed, the lack of methodologies to support the design of distributed applications makes the task of designers very difficult. The aim of our work is to contribute to the design of complete distributed applications, stressing first and foremost the quality of the final systems. To reach this goal we propose a software development environment based on a three-phased approach: analysis, design and construction. We will focus on the design and construction phases so as to define the architecture of the application, and to propose an implementation on a technical environment based on a distributed application model including the virtual node concept. Our approach uses the VDM formal method to support the design phase and the Conic distributed language and environment as the target for the implementation.

Keywords

Distributed application, programming in the large, reliability, formal specification, refinement, virtual node, module, port, communication.

1. Introduction

Today the software engineering community is convinced that there is a need to master the design of software. Developed software is becoming bigger and bigger and it is obvious that its complexity increases with its size. The maintenance costs is evaluated at 70% of the overall development cost. The aim of software engineering is clearly to find means to build quality software[21], and it is commonly accepted that the best ways to achieve this is particularly by abstraction, modularity, reusability[9].

However the recent development of workstations and communication networks proves that distributed programming is now a reality and the task of software designers is all the more difficult. The support available is mainly distributed operating systems, for instance: Amoeba [27], Chorus[24], Mach[1], parallel languages such as Ada, Occam, realtime languages like
Esterel[3] or distributed environments: Conic[17], Diadem, Dragoon[2]. Regarding methods, work is just started and is mostly algorithm-based; we quote for instance techniques of systolic array design as described in[23], or methods consisting in expressing a calculus problem thanks to a set of recurrence equations, which will be derivated and interpreted as a concurrent processes network[10]. Numerous approaches of rigorous program development exist, in particular program transformations[11], possibly assisted by program derivation support tools like SACSO[12] for instance. To our knowledge, few attempts have been made to distributed applications[22].

The aim of our work is to contribute to the design of distributed applications, proposing a rational approach applicable to complete applications, stressing the reliability of the final systems. This approach will be integrated into an open multiwindowing software engineering environment. It will be possible to refine a distributed application from its formal specification, stepwise transforming it by applying elementary tactics one after the other, or a succession of elementary tactics gathered into a strategy. This environment will allow to represent and manipulate the used objects in a graphic way. It will be totally open so as to accept and integrate the definition of new tactics or new strategies suggested by the designer.

Our contribution is part of a general stepwise design approach of distributed application. This approach is composed of three steps: analysis, design and construction. The goal of the analysis is to produce a first specification of the application from the requirements of the problem. From this model the design step leads to the architecture of the application. Eventually the construction step will effectively produce the code of the executable application.

The temptation is to jump directly from the application specification to an implementation on the technical target architecture, but we would be forgetting away a very determinant phase for quality of the application: the design step. Therefore we will concentrate on the design step (Fig.1) rather than the analysis and the low-level parts of construction. From an initial VDM formal model[13], the design step leads to the architectural structure of the application which complies to the underlying model of Conic distributed environment[17].

![Data transformations Groupings Implementation Operations rewriting](image)

**Fig.1 - Rational design steps**

From the architecture of the distributed application produced by the design phase, the construction phase will generate the code of the application itself, and determine its physical configuration. The choice of the technical target architecture is defined by its underlying concepts of modularity, abstraction and communication and its configuration facilities. It is not our goal to define a total implementation on the technical architecture, the configuration on the physical architecture or the execution control of the resulting applications are not our purpose, and we will propose straightforward solutions so as to illustrate the treated examples.

The design phase is based on data transformations. The techniques applied are the essential features of VDM, namely the specification refinement technique and the refinement correctness.