Foreword

Coordination models, languages and architectures

Paolo Ciancarini, Roberto Gorrieri

Dipartimento di Scienze dell'Informazione, Università di Bologna, Mura Anteo Zamboni 7, I-40127 Bologna, Italy

About coordination

The diffusion of multiprocessors and network technology with increasing bandwidth has fuelled the development of distributed computing and concurrent programming. Coordination languages are a new class of programming languages which offer a solution to the problem of managing the interaction among distributed and concurrent programs. Gelernter and Carrier0 used the term Coordination in the following context:

Programming = Computation + Coordination

They formulated this equation to explain the use of the coordination language Linda. Their point is that there should be a clear separation between the description of the components of the computation and the description of their interaction in the overall architecture of a software system. On the one hand, such a separation facilitates the reuse of code; on the other hand, the same patterns of interaction and software architecture occur in many different problems - so it becomes possible to reuse the coordination component as well!

Usually coordination languages are not general purpose programming languages; rather, they are often defined as language extensions or scripting languages and they are exclusively concerned with coordination issues.

In defining coordination languages there are a number of issues which must be addressed:
1. what is being coordinated?
2. what are the media for coordination?
3. what are the protocols and rules used for coordination?

Coordinated entities: The coordinated entities are usually active components - agents or processes. Coordination of agents should not require reprogramming of the agents; the coordination mechanism is a wrapper around the existing, independent agents. The agents may have been programmed in a variety of different programming languages.

Coordination media: In many coordination languages, coordination is accomplished via a shared data space that is a multiset of some data structures. In such models,
communication is *generative*: agents communicate by "generating" data in the shared space, this data is then available to any other agent that has access to the space — this contrasts with the message-passing paradigm of concurrency where communication is usually a private act between the participating agents. Coordination languages should let to control not only the active components, but also the coordination medium.

**Coordination rules:** The Linda proposal identifies a set of coordination primitives which may be used to access a shared data space — the primitives are normally implemented as library routines which are called from some host language such as C or Prolog. The semantics of such primitives can be exploited to optimize a coordination program with respect to a specific hardware architecture. In contrast to Linda, many of the recent proposals have been for rule-based languages; one consequence of this shift to a more declarative view of coordination is increased reasoning power. In either case the coordination "rules" provide a level of abstraction which hides much of the complexity of coordination from the programmer. Another extension to Linda which is increasingly studied is the concept of multiple tuple spaces, which allows to model in a simple way logical distribution and mobility of activities.

Roughly, coordination research concentrates on the following topics:

- coordination models, namely the study of semantics models and logics suitable to describe coordination languages and their concepts: coordination entities, media, and rules;
- coordination mechanisms and language design, namely the study of new coordination primitives, languages, and related programming tools like compilers and debuggers;
- coordination architectures and applications, namely the study of interaction structures, their specification and design into real distributed systems.

These topics were the subject of the ESPRIT Long Term Research Project 9102. The collections [1, 2] include some early papers on the theme of coordination languages. The collection [3] contains papers which were presented at the various Coordination project workshops. The project partners organised the first International Conference on Coordination Languages and Models, which was held in Cesena (Italy) in April 1996 [4]. The second conference was held in Berlin in September 1997.

**About this issue**

Following the first Coordination conference, two special issues were planned of the journals *Theoretical Computer Science* (guest edited by R. Gorrieri and C. Hankin) and *Science of Computer Programming*. We invited some selected participants from the conference, plus some members of Coordination project to contribute to this SCP special issue. The submissions were subjected to the usual reviewing process and eight papers were finally selected for inclusion in this special issue.

(CLF), a programming platform for coordination architectures made of distributed objects, whose theoretical roots are in the field of Linear Logic.

*The discrete time TOOLBus – A software coordination architecture*, by J.A. Bergstra and P. Klint, describes the formal specification and design of a coordinative software architecture which allows to build distributed applications by integration of separate applications.

*A logic for a coordination model with multiple spaces*, by P. Ciancarini, M. Mazza and L. Pazzaglia, introduces a TLA-based logic to reason on the software architecture of distributed programs specified using the PoliS coordination model. The logic is supported by a theorem proving environment based on the Larch prover.

*Structured Gamma*, by P. Fradet and D. Le Métayer, extends the coordination language Gamma with new constructs which give structure to the multiset – the basic coordination medium in Gamma. They also show how Structured Gamma can be used to specify software architectures.

*Distributed coordination with MESSengers*, by M. Fukuda, L.F. Bic, M.B. Dillencourt and F. Merchant, describes a coordination model based on autonomous, active messages which encapsulate mobile code.

*Coordination in the ImpUNITY Framework*, by H.J.M. Goeman, J.N. Kok, K. Sere and R.T. Udink, describes an extension of the UNITY logic, which allows to reason on the specification and coordination of software components written in different languages.

*Solving the Linda multiple rd problem using the copy-collect primitive*, by A.I.T. Rowstron and A.M. Wood, studies some new coordination mechanisms for Linda, which aim at solving the problem of implementing transactions over a tuple space.

*Laura – A service-based coordination language*, by R. Tolksdorf, describes Laura, a novel coordination language which redefines the tuple space coordination model for designing open systems based on the client/server architecture.

Three papers are from authors involved in the Coordination project, whereas five papers extend works published in the proceedings of the First Coordination conference. We thank the anonymous reviewers who helped us in the selection: their help was invaluable in putting together this special issue.

References