

Software Engineering for Distributed Systems

Paddy Nixon, Vinny Cahill
Distributed Systems Group,
Department of Computer Science,
Trinity College,
Dublin 2, Ireland
{pnixon, vjcahill}@dsg.cs.tcd.ie

Fethi Rabhi
Department of Computer Science,
University of Hull,
Hull, UK
F.A.Rabhi@dcs.hull.ac.uk

1 Motivation

Motivated by the unprecedented growth of the internet and associated new technologies such as Java and ActiveX, phrases such as *Net-Centric computing* and *write once, run anywhere applications* are becoming widely used. New applications are being developed which depend wholly, or at least significantly, on a distributed environment. Internet applications represent only one facet of distributed computing as a whole [1, 2]. Distributed applications occur in control systems, financial banking systems, and in offices and work places around the world. Nevertheless, the internet phenomena has brought into focus the difficulty in producing quality distributed software, and the need for new approaches to augment the software engineering process. For these reasons this mini-track investigates novel approaches which support the production of *quality* distributed applications.

2 Contributions

The papers presented are grouped into four related blocks emphasising the diversity and complexity of the subject area.

The papers by Chandy and Rifkin, Rakotonirainy *et al*, and Franke *et al* discuss new construction and composition models for distributed applications. Chandy and Rifkin show how internet applications can be composed in a structured manner. Rakotonirainy *et al* provide an architecturally-neutral semantic model for constructing open distributed systems. Franke *et al* describes how multiple development models can be composed in a uniform manner to produce a multi-view development environment.

Ensuring the correctness of distributed applications introduces a level of complexity not normally associated with their sequential counterparts. Four papers address this problem. Mazzocca *et al* discuss formal verification through the integration of Trace Logic and Petri Nets specifications to provide an intuitive and

modular notation. Kilgore and Chase introduce a novel debugging technique based on re-execution of programs. Kunz *et al* provide a method for managing the size of process-time diagrams and hence enable event tracing of large distributed systems. Mock *et al* introduce awareness services as a means of collecting information about the execution of a distributed application.

Two further papers present important development issues pertinent to distributed software systems. Performance is often a strong motivating factor for choosing a distributed solution. Parashar and Hariri introduce an interpretive solution to performance prediction and discuss the impact of this model on the development process. Takura visits another important topic and shows how communications software can be readily derived from state transitions rules.

Finally, some application areas are considered. Papers by Pepper and Sudholt, and Luksch consider scientific computing, and a paper by Deng *et al* considers distributed real-time systems. These papers give domain specific solutions to a subset of the problems encountered in distributed systems.

The construction of distributed software systems clearly brings new problems into the domain of software engineering as evidenced by the number and quality of submissions for this mini-track. The work presented here makes a significant contribution to alleviating these problems and provides a strong basis for future research in this area.

References

- [1] I. Gorton I. Jelly P. Croll and P. Nixon. Directions in software engineering for parallel systems. In B. Shriver H. El-Rewini, editor, *27th International Hawaii Conference on System Sciences*, volume II. IEEE Press, January 1995.
- [2] Cherri Pancake. Where are we headed. *Communications of the ACM*, 34(11):53-64, November 1991.