Aspect-oriented software engineering and component-based software engineering are finding increasing interest among both researchers and practitioners. The inception and adoption of these new technologies has been led by the drive to improve productivity, quality and reusability and reduce maintenance overheads, impact of customisations or extensions and the time to market. Aspect-oriented software engineering employs special abstractions known as aspects to separate crosscutting concerns throughout the software life cycle. Crosscutting concerns are features which cannot otherwise be cleanly encapsulated in one development artifact and are tangled over several artifacts. Special composition rules combine the aspects with artifacts (crosscut by features encapsulated by the aspects) with respect to reference points in the artifacts. These reference points are termed as join points. Separation of crosscutting features makes it possible to localise changes during maintenance, customisation and extension and helps improve productivity and quality. Some aspects can also be highly reusable e.g. domain specific aspects such as those encapsulating platform specific features.

Component-based software engineering comprises of two separate but related processes namely component engineering and application engineering. The former is concerned with the analysis of domains and development of generic and domain-specific reusable components while the latter involves application development using commercial off-the-shelf components (COTS) or components that have been developed in-house. Provided an effective risk management process is in place, this philosophy of constructing systems from reusable ‘parts’ promises a high degree of customisability and extensibility with increased productivity gains, accelerated time to market and lower development cost.

As these two technologies mature there is a significant interest and curiosity about their relationship with each other. This special issue includes three very interesting papers which explore this relationship and highlight the complementary nature of the two technologies. The application domains covered in the papers also form an interesting mix and include component-based distributed systems, embedded, real-time systems and software engineering environments.

Blair and her colleagues propose an aspect-oriented specification technique to support the specification of component-based distributed systems. The approach is motivated by the need to reason about the non-functional, quality of service (QoS) properties such as availability, dependability, security, etc. during the development of such systems. The authors focus on dynamic QoS management functions in general and monitoring and adaptation in particular. Tool support for the aspect-oriented environment is described. Support for synthesising QoS management components from particular aspects in the specification is also highlighted. The authors use a simple example of dynamic QoS management of an audio stream to describe the specification and verification of QoS management subsystems followed by synthesis of components that can be placed into a running system.

Rastofer and Bellosa highlight the lack of support for concepts such as execution time and resource usage in current component models; these concepts are of critical importance in distributed, embedded, real-time systems. They also discuss how these non-functional properties are influenced by the platform chosen for execution. Their proposed solution is a component model which separates platform specific issues of concurrency, synchronisation and distribution from the component functionality in a fashion similar to aspects. The behaviour of components is described using a path-based notation similar to use case maps. It is possible to deduce the behavioural description of an application from this path-based notation. The authors also describe the adaptation of components to an execution platform in their model and creation of real-time applications from these components.

The first two papers concentrate on different properties of distributed component-based systems. The third paper by Herrmann and Mezini focuses on a different domain: component-based software engineering environments and explores the role of aspect-like mechanisms to support configurability in such environments. The authors argue that, when constructing software engineering environments out of existing tools, the modularity, maintainability and extensibility of the resulting environment can be severely compromised if data model mismatches among the tools are bridged through data translators grafted into the implementation of individual tools. Their approach involves the use of explicit language constructs, called dynamic view connectors, to bridge mismatches in the data models. Dynamic view connectors are similar to aspects as they make it possible to separate the tool functionality from concerns of bridging data model mismatches. This improves the configurability and extensibility of the software engineering environment.

The three papers in this special issue are extended versions of selected papers from the Invited Session on Generative and Component-based Software Engineering held at the 4th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI), July 23–26, 2000, Orlando, Florida (USA). I wish to thank the SCI organisers especially Professor Nagib Callaos for making it possible to publish these papers in this special issue. I also wish to thank my co-organiser for the SCI 2000 invited session Jen Allanson and the program committee members and additional reviewers who helped review both the initial submissions and extended versions for this special issue. These include Mehmet Aksit, Don Batory, Gordon Blair, Lynne Blair, Alan Brown, Ulrich Eisenacker, Stan Jarzabek, Gregor Kiczales, Gerald Kotonya, Anthony Lauder, Nikolaos Parlavantzas, Elke Pulvermüller, Andreas Speck and Bedir Tekinerdogan.

Although the articles in this special issue provide a good insight into the complementary nature of aspect-oriented software engineering and component-based software engineering, there are several interesting and outstanding issues...
that are not covered here. Nevertheless, the contribution of these papers is significant as they have laid the foundation for bridging two highly promising, emerging software engineering technologies.

AWAIS RASHID

Dr Awaïs Rashid is a Lecturer at the Computing Department of Lancaster University in the United Kingdom. His principal research interests are in Aspect-Oriented Software Engineering, Object-Oriented Databases and Extensible Database Systems. He is particularly interested in combining different aspect-oriented approaches during system development and application of aspect-oriented techniques during component-based development. His other main research interest is in Aspect-Oriented Databases: providing support for aspects in database systems to reduce maintenance overheads and improve customisability and extensibility.