Software component engineering (SCE) is a broad term whose exact definition is currently a subject of considerable debate. SCE impacts a number of areas, ranging from distributed component environments (e.g., CORBA Component Model, COM, Java Beans) to more general issues regarding how to better modularize and deliver software regardless of context and domain. This minitrack encompasses two major thrusts: exploring meta-issues in defining a conceptual framework for SCE in general, as well as framework support for distributed component solutions.

The principal goal of this minitrack is to provide a forum that brings together people with a variety of perspectives and philosophies regarding software components. We are fortunate to have a collection of six key papers representing our two major thrust areas. In terms of defining a conceptual framework, Bergner, Rausch, Sihling and Vilbig provide an excellent introduction to some of the foundational issues related to software components in general. In particular, they propose a number concepts and definitions that might serve as a candidate conceptual framework for organizing the field. Fellner and Turowski provide a classification framework for business components. This work could also serve as a useful conceptual foundation, particularly with regard to organizing the various types of business components.

Two of the papers represent the application of specific design techniques to component design. In particular, Zhao and Kendall present an approach to component design that is based on role modeling. Mili, Dargham and Mili present a model of “view programming” that can be applied to the design and development of component-based systems. Both of these papers contribute to the conceptual foundation of the discipline by bringing with them a set of definitions and concepts that are applicable to component-based software.

Finally, two of the papers are particularly relevant to our second thrust area, that of distributed component systems. In particular, Mitra and Aggarwal propose a methodology and toolkit for the development of distributed component systems. They introduce a runtime environment for concurrent processes with a Linda-like shared memory concept. Giese proposes a approach to support distributed component system design based on contracts that place synchronization and other semantic requirements on the various components of the system.

The minitrack consists of a set of initial paper sessions as well as an open forum to support the discussion of issues in component-based software. Our hope is that the activities of this minitrack will result in evolution toward a more global and encompassing conceptual framework for software components.