Software evolution requires comprehension and modification of existing software systems, in which the system’s sheer size forces software engineers to work only with selected parts that are most relevant to the current task. It might also require performing such evolution-specific tasks as impact analysis, refactoring, and so on. The current literature offers few guidelines on how to teach software evolution.

At Wayne State University, an “Advanced Software Engineering” course teaches students tools and techniques for various evolution-process tasks and lets them practice evolution on software of considerable size. The students have practiced perfective software evolution on open source projects with sizes up to 68,000 lines of code without difficulty.

Service-oriented computing is the visionary promise of service-oriented computing that it will be possible to easily assemble application components into a loosely coupled network of services that can create dynamic business processes and agile applications that span organizations and computing platforms.

Such services will go well beyond simply exchanging information to accessing, programming, and integrating application services encapsulated within old and new applications.

Today, a service-oriented architecture is considered state of the art for service-delivery platforms. Such platforms for value-added services have evolved from the Intelligent Network and object-oriented programming interfaces to recent Web-services-based platforms. They’ve exploited the most recent information technologies to implement an open set of service components. Web 2.0’s recent emergence, meanwhile, has further pressured telecom companies to implement an open service market based on an open set of enabling services and service components.

Worldwide operations require global process modeling, coordination, and—at least since the Sarbanes-Oxley Act and Basel II—transparency. This puts enormous pressure on process management and its efficiency, compliance, reliability, and agility. Especially in large organizations, minimizing the total cost of ownership, controlling risk, and protecting the corresponding investment requires significant automation and standardization, often accompanied by radical reorganization.

The Centre of Innovation for Service-Centered Continuous Engineering adopts a holistic approach to closing the classical gap between business-driven requirements and IT-based realization, providing a seamless method and matching toolset based on the Java Application Building Center framework.

Establishing contracts for service composition remains challenging. Conducting extensive domain-specific research is a first step in addressing this problem. As knowledge develops, new industry standards in each policy domain will likely provide guidelines and patterns that will guide practitioners in various policy composition scenarios.

Semantic reasoning techniques hold the prospect of supporting automated, policy-rich service composition based on more meaningful service contracts.

Teaching Software Evolution in Open Source
 pp. 25-31
 Maksym Petrenko, Derys Poshyvanyk, Vaclav Rajlich, and Joseph Buchta

Evolution of SOA Concepts in Telecommunications
 pp. 46-50
 Thomas Magedanz, Niklas Blum, and Simon Dutkowski

Full Life-Cycle Support for End-to-End Processes
 pp. 64-73
 Bernhard Steffen and Prakash Narayan

Component Contracts in Service-Oriented Architectures
 pp. 74-80
 Francisco Carbera