integration. The connection between device integration and large, heterogeneous software systems will be examined. The technology itself will be presented and compared with other middleware, both distributed object and message-oriented, in the context of designing coherent, complex, scalable systems. I’ll close with a use case showing that the technology enables rapid, clean software practices, and how these in turn lead to a simplified, yet more powerful, notion of “work flow”.

Chair: Gail Kaiser, Columbia University, USA

4. Inter-Language Object Sharing with the Common Language Runtime: Infrastructure for MS .NET

Speaker: Jennifer Hamilton

Affiliation: Microsoft Corp.

Abstract: The Common Language Runtime (CLR) is language and platform-neutral, and provides the underlying infrastructure for the Microsoft .NET Framework. It consists of several components, including a garbage collector, class loader, metadata engine, just-in-time compiler, and debugging and security services. This talk will introduce the CLR, describe its design, and demonstrate how it enables inter-language object sharing.

Chair: Ric Holt, University of Waterloo, Canada

5. Collaborative Software Engineering

Speaker: Alan Brown

Affiliation: Catapulte Inc.

Abstract: Software development is a collaborative activity involving the sharing of ideas, artifacts, and knowledge among teams of engineers. Many elements are emerging to enhance the importance of collaboration in software development—geographically dispersed teams, more partnering within and across organizations, and leveraging of the Internet as a coordination infrastructure. In particular, the future of software development will be through leveraging the Internet as the base on which collaborative development services will be offered. To be successful requires that the next generation of software development environments be based on the “Three C’s”—community, collaboration, and coordination. This presentation examines the requirements for future software development environments, illustrates the kinds of services that must be offered, and examines the challenges that must be overcome in collaborative environments for building the next generation of enterprise scale software solutions. As a result, the presentation explores the current state of collaborative software engineering, analyses the currently available collaborative technologies, and establishes the current state-of-the-practice in collaborative software engineering.

Chair: Paul Sorenson, University of Alberta, Canada

6. AntiPatterns in Software Architecture

Speaker: Tom Mobray

Affiliation: World Wide Institute of Software Architects

Abstract: This presentation will provide an introduction to common AntiPatterns in Software Architecture, and their resolution through reference models. AntiPatterns are an intriguing variation of design patterns. AntiPatterns clarify ineffective software practices, as well as raise awareness of alternative solutions. In the context of software architecture, AntiPatterns can clarify the necessity of problem-solving skills revealed in software engineering reference models, such as, the Software Design Level Model (SDLM) and the Reference Model for Open Distributed Processing (RM-ODP). In particular, this presentation will discuss how traditional software engineering assumptions become serious AntiPatterns, when applied to distributed software architectures.

Chair: Margaret-Anne Story, Univ. of Victoria, Canada

7. Dependability of Embedded Systems

Speaker: John Knight

Affiliation: University of Virginia

Abstract: Embedded systems present especially difficult challenges in software engineering because they usually operate in real time on machines with modest resources and can cause extensive damage if they fail. In practice the interplay between systems and software plays a significant role in dependability and must be considered in embedded software development. In this presentation the basic issues of dependability as they apply to
embedded systems will be discussed, and several systems-engineering and software-engineering techniques for improving dependability will be reviewed. Topics to be covered include hazard analysis, fault-tree analysis, formal specification, design for safety, and verification.

Chair: Anatol Kark, National Research Council, Canada

8. Simple Object Access Protocol (SOAP) and Web Services

Speaker: Arthur Ryman

Affiliation: IBM Canada, Ltd.

Abstract: Simple Object Access Protocol (SOAP) is an exciting new technology for developing Web Services. A Web Service is a set of related application functions that can be programmatically invoked over the Internet. In contrast to traditional Web applications, Web Services are application building blocks that use Extensible Markup Language (XML) for information interchange. In this session you will learn about SOAP, Web Services, and related standards including Web Services Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI). These technologies will be demonstrated using the IBM XML and Web Services Development Environment which is available for free download from the alphaWorks Web site at http://www.ibm.com/alphaWorks/tech/WSDE.

Chair: Hausi Müller, University of Victoria, Canada

9. Auditing Legacy Systems for Security and Survivability

Speaker: Tom Longstaff

Affiliation: Software Engineering Institute, Carnegie Mellon University

Abstract: In our rapidly changing IT environment, it is essential that the security posture of all systems and networks are understood. This is extremely difficult when most of the systems that make up our infrastructures are legacy systems. Current state of the art techniques for auditing systems assume that expertise exists in all of the systems and platforms that make up the environment, when in fact many systems that are essential in the infrastructure predate any member of the auditing team. In this case, we show how use cases and intruder scenarios can be used to determine which systems are important from a security perspective and which services and properties of these systems must be verified. These techniques are part of the Survivable Network Analysis method, developed at the Software Engineering Institute to provide a measure of survivability for complex system architectures.

Chair: Prem Devanbu, Univ. of California, Davis, USA