Software Defined Radio
The Software Communications Architecture

John Bard, Space Coast Communication Systems Inc., USA
Vincent J. Kovarik Jr., Harris Corporation, USA

John Wiley & Sons, Ltd
Software Defined Radio
The Wiley Series in Software Radio aims to present an up-to-date and in-depth picture of the technologies, potential implementations and applications of software radio. Books in the series will reflect the strong and growing interest in this subject. The series is intended to appeal to a global industrial audience within the mobile and personal telecommunications industry, related industries such as broadcasting, satellite communications and wired telecommunications, researchers in academia and industry, and senior undergraduate and postgraduate students in computer science and electronic engineering.

Software Defined Radio
The Software Communications Architecture

John Bard, Space Coast Communication Systems Inc., USA
Vincent J. Kovarik Jr., Harris Corporation, USA
2 Operational Scenarios

2.1 Startup 22
2.2 Shutdown 26
2.3 Application (Un)Installation 28
2.4 Instantiate Application 30
2.5 Control Application 32
2.6 System Configuration 34

3 General Requirements and Services 37

3.1 Non-Functional Requirements 37
  3.1.1 General Requirements 38
  3.1.2 General Software Rules 39
  3.1.3 Hardware Architecture Requirements 39
  3.1.4 Interface Organization 40
3.2 Name Service 42
3.3 Event Service 44
  3.3.1 Event Types 46
3.4 Log Service 47
  3.4.1 Data Types 48
  3.4.2 Exceptions 48
  3.4.3 Types 50
  3.4.4 LogStatus Operations 54
  3.4.5 LogAdministrator Operations 56
  3.4.6 LogProducer Operations 58
  3.4.7 LogConsumer Operations 61
3.5 FileSystem 63
  3.5.1 Exceptions 64
  3.5.2 Types and Constants 64
  3.5.3 Types 66
  3.5.4 Operations 67
3.6 File 75
  3.6.1 Exceptions 76
  3.6.2 Attributes 77
  3.6.3 Operations 78

4 Foundation Interfaces and Data Types 83

4.1 TestableObject 83
  4.1.1 Exceptions 83
  4.1.2 Operations 84
4.2 PortSupplier 86
  4.2.1 Exceptions 87
  4.2.2 Operations 87
4.3 LifeCycle 88
  4.3.1 Exceptions 88
  4.3.2 Operations 88
4.4 PropertySet 89
  4.4.1 Exceptions 89
  4.4.2 Operations 90
4.5 Resource 92
  4.5.1 Exceptions 92
  4.5.2 Attributes 93
  4.5.3 Operations 93
4.6 ResourceFactory 95
  4.6.1 Exceptions 95
  4.6.2 Attributes 96
  4.6.3 Operations 96
4.7 Port 99
  4.7.1 Exceptions 101
  4.7.2 Operations 102

5 Devices and the Device Manager 105

5.1 Introduction 105
  5.1.1 SCA Device Abstraction 106
5.2 Device 108
  5.2.1 Exceptions 109
  5.2.2 Types and Constants 109
  5.2.3 Attributes 110
  5.2.4 Operations 118
5.3 LoadableDevice 121
  5.3.1 Types 122
  5.3.2 Exceptions 123
  5.3.3 Operations 123
5.4 ExecutableDevice 127
  5.4.1 Types and Constants 127
  5.4.2 Exceptions 128
  5.4.3 Operations 130
5.5 AggregateDevice 134
  5.5.1 Types and Attributes 134
  5.5.2 Operations 134
5.6 DeviceManager 135
  5.6.1 Types 136
  5.6.2 Attributes 137
  5.6.3 Operations 140

6 Domain Management 151

6.1 DomainManager 151
  6.1.1 Types 151
  6.1.2 Exceptions 153
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1.3 Attributes</td>
<td>155</td>
</tr>
<tr>
<td>6.1.4 DomainManager Instantiation</td>
<td>157</td>
</tr>
<tr>
<td>6.1.5 Operations</td>
<td>158</td>
</tr>
<tr>
<td>6.2 FileManager</td>
<td>178</td>
</tr>
<tr>
<td>6.2.1 Types</td>
<td>180</td>
</tr>
<tr>
<td>6.2.2 Exceptions</td>
<td>180</td>
</tr>
<tr>
<td>6.2.3 Operations</td>
<td>181</td>
</tr>
<tr>
<td>6.3 The ApplicationFactory</td>
<td>183</td>
</tr>
<tr>
<td>6.3.1 Exceptions</td>
<td>184</td>
</tr>
<tr>
<td>6.3.2 Attributes</td>
<td>184</td>
</tr>
<tr>
<td>6.3.3 Operations</td>
<td>185</td>
</tr>
<tr>
<td>6.4 Application</td>
<td>192</td>
</tr>
<tr>
<td>6.4.1 Types</td>
<td>192</td>
</tr>
<tr>
<td>6.4.2 Attributes</td>
<td>193</td>
</tr>
<tr>
<td>6.4.3 Operations</td>
<td>195</td>
</tr>
<tr>
<td>6.4.4 General Requirements</td>
<td>199</td>
</tr>
<tr>
<td>7 Operating Environment Security</td>
<td>201</td>
</tr>
<tr>
<td>7.1 Core Framework Security Requirements</td>
<td>201</td>
</tr>
<tr>
<td>7.1.1 Application</td>
<td>201</td>
</tr>
<tr>
<td>7.1.2 ApplicationFactory</td>
<td>202</td>
</tr>
<tr>
<td>7.1.3 DomainManager</td>
<td>203</td>
</tr>
<tr>
<td>8 Certification</td>
<td>205</td>
</tr>
<tr>
<td>8.1 Certification Process</td>
<td>205</td>
</tr>
<tr>
<td>8.2 Operating Environment Certification</td>
<td>206</td>
</tr>
<tr>
<td>8.2.1 OE-1</td>
<td>206</td>
</tr>
<tr>
<td>8.2.2 OE-2</td>
<td>208</td>
</tr>
<tr>
<td>8.2.3 OE-3</td>
<td>209</td>
</tr>
<tr>
<td>8.3 Waveform Assessment and Certification</td>
<td>210</td>
</tr>
<tr>
<td>9 The Domain Profile</td>
<td>215</td>
</tr>
<tr>
<td>9.1 Overview</td>
<td>215</td>
</tr>
<tr>
<td>9.2 SCA Domain Profile XML</td>
<td>215</td>
</tr>
<tr>
<td>9.3 Domain Profile Data Types</td>
<td>218</td>
</tr>
<tr>
<td>10 Base Descriptor Files</td>
<td>219</td>
</tr>
<tr>
<td>10.1 Properties Descriptor</td>
<td>219</td>
</tr>
<tr>
<td>10.1.1 Simple</td>
<td>219</td>
</tr>
<tr>
<td>10.1.2 Simple Sequence</td>
<td>222</td>
</tr>
<tr>
<td>10.1.3 Struct</td>
<td>222</td>
</tr>
<tr>
<td>10.1.4 Struct Sequence</td>
<td>224</td>
</tr>
<tr>
<td>10.1.5 Test</td>
<td>224</td>
</tr>
</tbody>
</table>
10.2 softpkg
   10.2.1 title
   10.2.2 author
   10.2.3 description
   10.2.4 propertyfile
   10.2.5 descriptor
   10.2.6 implementation
10.3 Software Component Descriptor
10.4 Device Package Descriptor

11 Device Configuration Descriptor

11.1 Overview
11.2 deviceconfiguration
   11.2.1 description
   11.2.2 devicemanagersoftpkg
   11.2.3 componentfiles
   11.2.4 partitioning
   11.2.5 connections
   11.2.6 domainmanager
   11.2.7 filesystemnames

12 The Domain Manager Descriptor

12.1 Overview

13 The Software Assembly Descriptor

13.1 Overview

PART III Building an SCA-Compliant System

14 The POSIX Operating System

14.1 An Operating Environment
14.2 Linux 2.6 Kernel
   14.2.1 Unavailable POSIX Calls
   14.2.2 More Unavailable POSIX Calls

15 POSIX Threads

15.1 The Thread Object
15.2 Un-named Semaphores
15.3 Mutex Variables
15.4 Thread Attributes
15.5 Conditional Variables
15.6 Less Interesting Thread Calls
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.7</td>
<td>Properties</td>
<td>373</td>
</tr>
<tr>
<td>18.8</td>
<td>Manipulating Ports</td>
<td>378</td>
</tr>
<tr>
<td>18.9</td>
<td>Summary</td>
<td>378</td>
</tr>
<tr>
<td>19</td>
<td>An SCA-compliant Application</td>
<td>383</td>
</tr>
<tr>
<td>19.1</td>
<td>Hello World Legacy Application</td>
<td>383</td>
</tr>
<tr>
<td>19.2</td>
<td>Legacy Hello World SPD</td>
<td>388</td>
</tr>
<tr>
<td>19.3</td>
<td>HMI Applications</td>
<td>391</td>
</tr>
<tr>
<td>19.4</td>
<td>Shutting Down</td>
<td>396</td>
</tr>
<tr>
<td>19.5</td>
<td>An SCA-compliant Hello World Application</td>
<td>397</td>
</tr>
<tr>
<td>19.5.1</td>
<td>An SCA-compliant Terminal Device</td>
<td>397</td>
</tr>
<tr>
<td>19.5.2</td>
<td>Domain Profile for Terminal Device</td>
<td>405</td>
</tr>
<tr>
<td>19.5.3</td>
<td>An SCA-compliant Talk Application</td>
<td>409</td>
</tr>
<tr>
<td>19.5.4</td>
<td>Multi-threaded Servant</td>
<td>414</td>
</tr>
<tr>
<td>19.5.5</td>
<td>Talk Application XML</td>
<td>417</td>
</tr>
<tr>
<td>19.5.6</td>
<td>Modifications for Minimum CORBA Compliance</td>
<td>423</td>
</tr>
<tr>
<td>19.5.7</td>
<td>Concluding Remarks</td>
<td>424</td>
</tr>
<tr>
<td>Appendix A</td>
<td>Mandatory POSIX Calls</td>
<td>427</td>
</tr>
<tr>
<td>Appendix B</td>
<td>References to Part III</td>
<td>429</td>
</tr>
<tr>
<td>Index</td>
<td></td>
<td>431</td>
</tr>
</tbody>
</table>
Disclaimer

The viewpoints, perspectives, and opinions expressed in this book are solely those of the authors and do not represent or reflect any position, opinion, or interpretation on the part of Harris Corporation, Space Coast Communication Systems, the Joint Tactical Radio System (JTRS) Joint Program Executive Office (JPEO), or any other government or industry organization.
Acknowledgments

Indeed, this has been a wonderful journey for Vince and I, but it hasn’t been nearly as pleasant for our loved ones. To this end, I dearly want to thank Pam, Amela, and Amanda for their long suffering and understanding. They stopped believing me when I said ‘I’m almost done’, but now, for sure, I’m done.

I am grateful to Dr. Sam Aslam-Mir for providing source material used in Chapter 16. His knowledge and pre-eminence in the CORBA community has added integrity and depth to an otherwise hackers approach to CORBA.

John Bard

First, I would like to thank John for the invitation to collaborate on this book. I would also like to thank Eric Held for providing valuable comments and Lance Starr for his support in developing the Domain Management ToolKit (dmTK®), an implementation of the SCA. Finally, a sincere thank you to my wife Madeline for her support and understanding throughout this process.

Vincent J. Kovarik, Jr.
When first approached by Wiley a few years back to write a book on software radio, I immediately knew that the topic could not be condensed to fit within a single volume – thus was conceived the Wiley book series on Software Defined Radio.

The early volumes addressed the emergence, concepts and international activities, and the technological foundations – radio, baseband and software. These were followed by a volume describing European research on software radio architectures and systems, from the mobile telecom industry perspective, and then one addressing the specific application of software radio baseband technologies to the emerging 3G marketplace.

This latest volume, by John Bard and Vince Kovarik, represents an essential and in many ways a long overdue element of the series, addressing as it does the Software Communications Architecture, or SCA, which lies at the heart of the world’s largest software radio project, the US JTRS programme and which had its own origins a decade or more ago.

The SCA is a term that many have heard, but few truly understand. John Bard and Vince Kovarik most definitely do not fit that description – indeed, they have spent many years now not simply developing their own understanding and contributing to the development of the SCA, but supporting others in applying the SCA to real world implementations. Thus, this volume is very much a book by practitioners, for practitioners. The book is effectively structured into three major sections – the first addressing the operating environment, the second focusing on the domain profile and finally a substantial set of chapters on building an SCA-compliant system.

Whilst originating within the context of the JTRS programme and the defence industry, the SCA has potential applicability beyond – this is an area which has yet to be commercially explored. Existing SCA practitioners have to date been so busy and focused on the defence requirements and contracts, that few experts have as yet had time and opportunity to begin to apply it in the commercial domain. Perhaps availability of this volume may contribute to the opening of such doors.

I would like to conclude this foreword by congratulating John and Vince on this epic work. When I first proposed to John that he consider writing a volume on the SCA he wisely said he would think about it and then, very quickly, proposed Vince as his co-author. At that time he knew, far better than I, the true scale of the task, one that has not been made any easier by the way in which the SCA has continued to dynamically evolve. That John and Vince have chosen to make the time to write this book – from personal experience I know that one doesn’t find time for such things – is a service to the industry for which many will be grateful; their book will serve as a foundation for many engineers in the coming years. Thank you John & Vince – well done.

Dr. Walter Tuttlebee
Chief Executive, Mobile VCE
Preface

Over the past decade radio system design has seen an inexorable march towards more of the waveform signal processing being performed digitally. As Moore’s law continues to push the capabilities of the General Purpose Processor (GPP), the processing power of the Digital Signal Processor (DSP), and the Field Programmable Gate Array (FPGA), this trend will continue to accelerate along with the power of these devices. The natural consequence of this trend is that more of the radio signal processing is being performed by software.

Although the use of software to perform more of the core radio functions has increased dramatically, each radio manufacturer developed solutions that differed in their architecture and implementation. Therefore, radio systems became more flexible as more capabilities were provided via software, and each implementation was unique. So, although the characteristics of the radio system could be changed through software, there was still little commonality in the control structure and management architecture across radio systems.

This lack of common management and control architecture was a significant problem in the military and public safety sectors. In these sectors, special-purpose radios were the norm rather than the exception. These radios were typically limited to a small set of capabilities or waveforms (or sometimes just one). Also, a significant number of legacy radios were hardware-based and, consequently, could not be re-configured without physical modifications or re-design. Those radios that were software-based were closed to software implemented by sources other than the original manufacturer. Compounding the problem was the fact that the radio system could not be managed, configured, or controlled using a consistent set of interfaces and protocols. So, when multiple radio sets from different manufacturers came together in response to some coordinated exercise, military operation, or disaster, the radios did not interoperate and could not be easily reconfigured to do so.

To address this problem, the United States government initiated a series of programs leading towards the specification of a common software infrastructure for software defined radios. The initiative started in the mid-1990s and evolved into the Software Communications Architecture (SCA). Although there have been prior radio infrastructures and architecture, the SCA is the first such specification that represents the combined contributions of many of the key radio system manufacturers for the United States government.

Audience

This book focuses on the SCA architecture; the use, issues, and benefits associated with developing a radio system in compliance with the specification. The book is intended to
provide practical information on building an SCA-compliant system along with historical and conceptual background information to help fill in the gaps between the intent of the specification and the practice.

This book does not provide instruction on how to construct an implementation of the SCA specification, i.e. a Core Framework. Rather, it is intended to provide guidance on how to use a Core Framework implementation to build a software radio system, provide an SCA-compliant hardware component for a radio system, or deliver a waveform for use within an SCA radio system.

Consequently, this book is intended for anyone who has the need to design, develop, support, or understand hardware or software in an SCA-compliant software radio system. It provides information, background data, and interpretations of requirements presenting design tradeoffs and decisions that must be addressed in fielding an SCA-compliant system.

Scope

As with any technical book that addresses a dynamic and changing technology, certain decisions and tradeoffs were made regarding the scope. At the time this project was initiated, SCA version 2.2 was the current version of the standard and was chosen as the baseline for this book. SCA 2.2 continues to be the SCA version that is the contractual baseline for a number of on-going contracts.

However, the SCA has continued to evolve. In April 2004, SCA 2.2.1 was released. SCA 2.2.1 provided several clarifications and corrections to issues identified in SCA 2.2. Then, in August 2004, SCA 3.0 was released. SCA 3.0 was the first SCA release to attempt to address issues related to waveform component construction and portability in the realm of DSPs and FPGAs. These processors, although an integral part of a software radio, introduce coding, transport, and interface requirements not addresses in prior SCA specifications. Upon its release, however, it was apparent that SCA 3.0 had a number of shortcomings. Primarily, although it began to address the issues noted above, SCA 3.0 did not address portability issues related to DSP and FPGA processors substantively enough to provide a workable set of requirements and guidance for these processors. In the area of Core Framework functionality, there was little change to the baseline requirements.

As the final version of this book was being prepared for the publisher, version of SCA 2.2.2 was released. SCA 2.2.2 further refined the specification, corrected errors and omissions, and provided clarifications to the specification. Where possible, notes and information regarding key aspects of SCA 2.2.1 and 2.2.2 have been incorporated into this book.

Conventions Used in This Book

The following typography and terminology are used throughout the book to help the reader.

Source Code

Source code is presented using Courier New font, as shown below.

```c
int add1( int number) {
    return number++;}
```
Preface

Terminology

In this book, the term **SCA-compliant** will be used to denote that a part or the whole system adheres to the requirements specified in the SCA specification. Hence, it is compliant with the specification. The term **SCA-certified** or just **certified** is used only when referring to a system or component that has passed the suite of tests performed by JTRS Test Application (JTAP) tool.

**Unified Modeline Language (UML)**

Diagrams of software architecture and design are drawn using the UML graphical modeling language developed by the Object Management Group (OMG). Information on the UML standard can be found at http://www.uml.org/.

Organization

The book is divided into three parts. Part I focuses on the Operating Environment (OE). The OE forms the common software infrastructure for implementing a software defined radio that is SCA-compliant. General background on software defined radio systems, the origins and history of the SCA evolution, and organization of the SCA specification and components are provided. The SCA is discussed from the perspective of a hardware supplier, waveform developer, radio system integrator, or other individual or corporation that must utilize and adhere to the SCA requirements. Topics include the OE components, the requirements for each component, a discussion of the essential concepts or rationale behind the component along with discussion of any pros and cons, and a section on the certification process.

Part II discusses the Domain Profile. The Domain Profile describes the hardware and software that form an SCA-compliant radio system and the applications that are hosted on the system. A set of eXtensible Markup Language (XML) files are used as a platform-neutral language to describe the hardware and waveform application components, their composition, and the underlying SCA-compliant platform.

Part III provides code examples that apply and use the capabilities of the SCA as described in the first two parts. The first offering is an in-depth view of the SCA Application Environment Profile (AEP) including code examples for the use of POSIX threads. This is followed by an examination of the OMG’s CORBA specification. Finally, a set of straightforward examples are presented with explanation and commentary to help understand and visualize the content presented in the first two parts of the book.

Additional Material and Author Contact

Additional reference material may be found at the website for the book hosted by the publisher. For more details, please visit the URL http://wiley.com/go/bard.

Although every effort has been made to ensure the accuracy of the content of this book, there may be errors. If you happen to find any errors or simply wish to provide comments and suggestions, you can contact the authors at vkoivarik@acm.org and jbard@spacecoastcomm.com.
In Part I, the Software Communications Architecture (SCA) specification is presented. The aim of this part is to provide an overview of the specification with specific discussions related to the interpretation and functional behavior of the SCA requirements. The intent is that this part can be utilized as a self-contained reference document for the SCA specification, with annotations and background information. As noted in the Preface, the objective of this book is to provide information on the understanding and use of an SCA Core Framework.
1

Introduction

The fundamental objective of the SCA is to provide a common software infrastructure for managing radio systems. Although software comprises a significant part of most recent radios – thus enabling new capabilities and functions to be added to the radio at some future times – the software is loaded and controlled through proprietary mechanisms and each radio manufacturer typically employs a unique infrastructure or architecture. A software defined radio, as interpreted here, refers to a class of radios, the capabilities of which are not simply provided by software but utilize an infrastructure that supports interchangeable components as well as functionality.

This chapter provide background information regarding the SCA. The SCA specification describes a collection of components, the configuration of the components, and the assembly of the components into a functional waveform application on a radio system. Taken together, these form an infrastructure for defining and constructing a software defined radio system.

1.1 Software Radios

Figure 1.1 illustrates the abstraction space of bandwidth versus waveform abstraction. At the lowest level is a set of hardware that provides the actual processing of the waveform and support software. The processing is provided by one of four options, General Purpose Processor (GPP), Digital Signal Processor (DSP), Field Programmable Gate Array (FPGA), and Application Specific Integrated Circuit (ASIC). The ASIC is typically not considered part of the solution set within a software radio because, once programmed, it cannot be modified after deployment – one of the fundamental tenets of a software radio.\footnote{This does not necessarily imply that an ASIC cannot be applied within a software defined radio. It is the authors’ opinion that, given certain circumstances and architectural approaches, it is feasible to integrate ASICs within the design of a software defined radio. Such a design would need to support the ability to interchange flexibly functional components of the waveform processing implemented in ASICs. This would require some mechanism such as the ability to call algorithms implemented on an ASIC as though it were a function call.}

The aim of Figure 1.1 is to illustrate the two orthogonal perspectives of software radio design. The waveform design starts as a set of requirements, simulation, mathematical
model, or some other conceptual representation. As the waveform progresses from design to implementation, the capabilities of the waveform, in terms of throughput and capacity, typically drive the implementation to a high-level language for deployment on a GPP or DSP. Higher throughput demands drive the deployment towards an FPGA or an ASIC.

The GPP processor typically provides the management and control services for the system. Overlaid on top of the processor is an operating system and, integrated with the operating system, is a collection of software that provides the run-time infrastructure for the radio set. The infrastructure, in SCA terms, is called the Core Framework. On top of the Core Framework sits the waveform and other applications.

1.1.1 Software Radio Aspects

A software radio system can be viewed through one of four perspectives or aspects. Each aspect forms a functional grouping of objects and services provided by the radio system. Illustrated in Figure 1.2, these aspects are:
• **Hardware** – This aspect describes the physical set of devices and components that comprise the radio set.

• **Software** – This aspect defines the set of services and interfaces through which all waveform applications must interface to the underlying hardware.

• **Application** – This aspect defines the application and service layer. All waveforms and common services execute in this aspect.

• **User** – This aspect is the view through which the user interacts with the radio set. There are two basic modes of interaction within this aspect. The user is either performing radio control operations, e.g. setting system parameters, or performing application control and data transfer, e.g. setting the gain parameter for a specific waveform instance.

The SCA can be viewed as one realization of the Software Infrastructure aspect with some parts within the Applications and Services aspect. It defines a logical infrastructure for management and abstraction of physical hardware components, a standard set of abstractions for software components that form the digital processing portion of a waveform implementation, general services available for use by the system, and a set of common interfaces for managing, deploying, and configuring waveform applications within the system.
1.2 The Software Communications Architecture

Any new concept or technology has a learning curve associated with it and the SCA is no exception. The SCA defines a software infrastructure for the management, control, and configuration of a software defined radio. It does not mandate any specific architecture, design, or implementation for the radio system hardware or waveform application. Before launching into the detailed discussion of the SCA, it is advisable to spend a short bit of time providing some background data and explanation on what the SCA is, and is not, the history of its evolution, and the reasons why you would (or would not) want to apply the SCA to your system.

The SCA is based on several related technologies: Object-Oriented (OO) techniques in software engineering, the Common Object Request Broker Architecture (CORBA), and the CORBA Components Model (CCM). Object-oriented languages have been around for a number of years from Simula in the late 1960s, Smalltalk and Flavors in the early 1980s, to current object-oriented languages such as C++, Python, Ruby, and Java, to name a few.

As systems evolved towards distributed architectures and a client-server model, CORBA evolved as an industry standard for describing the interfaces provided or used by two components using a pseudo-code called an Interface Definition Language (IDL). IDL provided the means for specifying the available interfaces and, through the IDL ‘compiler’, generated source code that is compiled into each of the applications. The code generated includes the support routines necessary to support remote procedure calls between processes on the same computer and between computers, i.e. in a distributed environment. Thus, the developer was freed from the drudgery of writing low-level, inter-process communications code and, more importantly, CORBA code built by one individual could interoperate with code built by another individual, the only requirement being that both the author of the client application and the server application use the same IDL. This was an important step forward in the ability to develop modular software while encapsulating the internal logic and requiring only that each of the developers agree on a set of IDL.

Although the CORBA technology provided several important advances, it became apparent that the mechanism by which systems were deployed was still dependent on manual configuration. The CCM evolved to address the need for specifying the requirements for deploying a set of application software by describing what resources were required to deploy the system successfully on a set of hardware. The method for describing the components of a system and the related deployment requirements is through a set of eXtensible Markup Language (XML) files. XML is a text-based language that utilizes tags to define items, their attributes, and values. This CCM XML was the genesis of the SCA Domain Profile XML.

With this brief summary of background information and foundation technology as a backdrop, the next sections provide a summary of what the SCA is, is not, why you would (or would not) want to use it, and a brief history of its evolution.

1.2.1 The Evolution of the SCA

The United States military was (and is) facing an increasingly critical need to support communications for multiple missions, rapid deployment, diverse mission scenarios and objectives, increased interoperability, and to reduce the cost of operations. One of the