Preface to: Component Reuse in Software Engineering (C.R.u.i.S.E.)

Welcome to Component Reuse in Software Engineering (C.R.u.i.S.E.), a product of the Reuse in Software Engineering (RiSE) Group. RiSE has taken on the formidable task of studying and establishing a robust framework for software reuse, and this book is a milestone achievement in that mission. The following pages contain an extremely thorough review and analysis of the field of reuse by collecting and organizing the major works into a cogent picture of the state-of-our-art.

However, this book is much more than just a very thorough compendium of reuse research. CRuiSE actually traces the *history* of our field from the nascent thoughts credited to McIlroy in 1968 right up to the present. In taking the reader through this history, CRuiSE touches on all the key phases in the evolution of reuse, from library systems, organizational issues, domain analysis, product lines, component-based engineering, to modern architectures for reuse. In each phase, the leaders in our field have poised numerous concerns and problems that remain to be solved. CRuiSE has retained the questions as posed by the original researchers, but also adds new analysis and a current perspective. While it is impossible to cite every single source, the authors have very effectively summarized and analyzed the important works, placing them in context along with the other related works at the time.

One thing that CRuiSE will immediately impress upon the reader, as it did me, is the amazing amount of literature and accumulated knowledge that exists in reuse. It is refreshing and with a bit of nostalgia that I again review the landmark articles and read about the first large-scale, practical reuse programs at industry-leading companies such as Hewlett-Packard, Motorola, and IBM, where I began my real hands-on experiences in this exciting field. I particularly like the use of timelines throughout the book to show the progression of the issues in these articles and relationship of the major accomplishments.

In addition to this evolutionary progression, CRuiSE analyses the omnipresent issues that challenge the successful introduction of reuse into every organization. These issues range from best practices in development processes, component certification, to library system search and retrieval. Of course the book presents metrics for determining both the value of reuse and the level of reuse practiced by an organization. This topic is near and dear to my heart because of the important role that metrics play in software development and even in helping to encourage the practice of reuse. In this regard, CRuiSE is equally valuable to the practitioner seeking to institute a reuse program in their company or organization.

CRuiSE not only faithfully traverses the history and past issues surrounding reuse, but the book also gives a thorough analysis of modern practices. There is an entire chapter dedicated to Component-Based Software Engineering (CBSE), which I believe is a key technology and mind-set that organizations must adopt if they really want to use assets in more than one place. The book extends CBSE into one popular modern embodiment of CBSE; namely, Service Oriented Architectures (SOAs). Whereas in the past, reuse has often been stymied by competing technologies (such as COM, CORBA, RPCs, etc.), the classic SOA leverages the worldwide acceptance of web technology and open protocols. So rather than a plethora of incompatible standards, suddenly we have universal adoption of a common architecture and development process. The future of reuse has never looked so bright!

Although reuse is, strictly speaking, the use of un-modified software assets, this book also discusses the important relationships between re-engineering and reuse. Re-engineering, often referred to as “white box reuse,” might reduce up-front development costs but, as many of us know, changing any part of a component can often cost more than just starting from scratch. Nonetheless, there are times that modifying an existing component becomes unavoidable. CRuiSE nicely summarizes the issues that arise in these situations, to include version control problems, reduced quality, and the proliferation of software that needs to be maintained.

CRuiSE is a must-read for students and researchers prior to beginning any new work in the field. Why? Because simply studying the state-of-the-art is not enough to advance the field. To move forward, researchers need to truly understand the string of successes and failures that shaped where the field is today. Likewise, the wise practitioner would have this book handy to understand the rationale behind a "best practice" that might be questioned as he or she rolls out a reuse program for their company. Quite simply, the history of these findings, failures, and partial successes determines the reason *why* we do things the way we do.

In conclusion, CRuiSE makes one message very clear:

Software Reuse depends on systematic processes and tools.

In other words, reuse doesn't happen by accident. The research described in this book focuses on issues ranging from technology to management and even to human behavior. Each issue has enjoyed the focus of many experts, and with the benefit of lessons and detailed information contained in CRuiSE, you will much better understand the systematic processes and tools that *you* need for success in your organization.

Dr. Jeffrey S. Poulin
jeffrey.poulin@lmco.com
Lockheed Martin Distribution Technologies
Owego, NY