MORALE METHODOLOGY GUIDEBOOK
Preface

Problem: Nobody reads prefaces
Solution: Call the preface Chapter 1.
New problem created by solution: Chapter 1 is boring.
Resolution: Throw away Chapter 1 and call Chapter 2 Chapter 1.

—Donald Gause and Gerald Weinberg

Most software engineering books have it wrong. They give a thorough description of the steps in creating a software system, from initial conception through requirements gathering and analysis, on to design, implementation, and testing. Some even discuss delivery and post-deployment maintenance. But most software development does not work that way. Most software development is concerned with correcting, adapting, and enhancing existing software. This book tells the software development story from the point of view of software evolution—how a software system can grow and change in a controlled way over its lifecycle, how design deterioration can be avoided, how future decision making can take effective advantage of the analyses and design decisions made while developing previous versions, how the impact of changing requirements on the underlying system architecture can be assessed and used to avoid costly mistakes, and how the legacy value of legacy software can be understood and leveraged.

This book is for software engineers. A software engineer need not be a programmer. A software engineer is anyone who works on a software development project and whose interests go beyond the purely technical concerns of programming languages or test cases. A software engineer is concerned with the *process* of software development as well as the *product* being developed. A software engineer is concerned with the success of a product during all phases of its development and throughout its lifetime. Software engineers include software managers who have to make tough decisions like buy-versus-build, scrap-versus-renovate, slip-versus-ship.

The book assumes that you are engaged in software development as a day-to-day activity, that the software you are developing is large enough to require a team of developers, and that the software is intended to last long enough that the long-term benefits of documentation, version control, and configuration management are appreciated. Although we define our terminology before we use it, we also assume that your are fluent with the language of software development, either as the result of formal education in computer science or due to the hard-won experience of actually fighting software battles from the front lines.

The book is structured as a series of guidebooks, each covering a different aspect of the software evolution process. Each guidebook describes in concrete terms how to effectively engage in its part of the software development process. And each guidebook presents a complete, worked example of applying its technique to a real software evolution problem. The worked examples all consider the same underlying software system, the Mosaic world wide web browser. Mosaic has gone through many versions over several years. Each of the guidebooks looks at the evolution of Mosaic from version 2.4 to version 2.5 and, in particular, how a specific change made for version 2.5—the addition of user-configurable viewers—can be analyzed, judged, and implemented in a controlled way.

The techniques described the guidebooks constitute the MORALE software evolution process. *MORALE* is an acronym for **M**ission **Or**iented **A**rchitectural **L**egacy **E**volution. *Mission oriented* refers to MORALE's assumption that the changes being made to a system are fundamental to the system's purpose and users, rather than being a purely technical change such as adapting the program to a new version of the hardware on which it runs. *Legacy* refers to the fact that evolution of an existing version is being considered and that determination of the most effective way to use its resources is central to planning the new version. And *architectural* emphasizes that the alterations being contemplated may have profound effects on the basic structural and coordinational principles upon which it has been build.

The book begins with an overview of MORALE and the Mosaic case study. The techniques from each guidebook are introduced and how it deals with the evolution of Mosaic is summarized. There are five guidebooks. The first talks about mission-oriented requirements elicitation and analysis. It also introduces a theme that pervades MORALE—scenarios. Scenarios are sequences of activities that together describe one aspect of system execution behavior. Scenarios are also central to the second guidebook, which presents a way to analyze the effect of changes on a software architecture. Changes to a system architecture are costly, and unexpected architectural changes can spell doom to an otherwise successfully managed project. The third guidebook is concerned with legacy-system understanding, in particular, architectural extraction. That is, how can the underlying system components and their interactions be determined, even in the face of out-of-date or incomplete documentation, deteriorating structure, and overwhelming size and complexity.

Even if the mission-oriented requirements can be determined, the architecture extracted, and the impact of the former on the latter determined, there is still the question of how to actually make the change. The fourth guidebook is concerned with planning for these changes. In particular, it considers the question of how the system's overall goals are realized in its functional components and how changes to those goals affect the components and their interaction. The fifth guidebook is concerned with a specific kind of change, change to a system's user interface. Not only is effective user interface design a difficult and time consuming process, but its result often determine the ultimate customer acceptance of a product. This guidebook describes a technique for analyzing the user interface of an existing system version and transforming it to conform to its new requirements.

Following the guidebooks are several useful appendices. The first describes the MORALE tool suite, Esprit de Corps, as an illustration of how some of the techniques described in the guidebooks can be supported by automated tools. The tools range from an information system for managing scenarios, to a program analyzer, to a graphical visualization tool for architectures, to two knowledge-based tools for performing, respectively, program adaptation and user-interface migration. The second appendix is a glossary, collecting and synthesizing terminology from all of the guidebooks. References, however, are not collected, appearing instead at the end of each of the chapters.

MORALE itself evolved from the work of its individual contributors. Some of the work described here derives from earlier written descriptions co-authored by colleagues who are not part of the MORALE project. Nevertheless, we wish to express our appreciation to them both for their contributions and for their permission to reprint their work. Individual acknowledgment appear in each of the chapters. MORALE first took its current form in a proposal to the Defense Advanced Research Projects Agency (DARPA) of the U.S. Department of Defense. MORALE has been funded by DARPA since July 1996. The MORALE project is part of DARPAs Evolutionary Design of Complex Software program (EDCS) with John Salasin as its program manager. We would like to thank John, not only for his financial support but also for his constant encouragement to make our ideas tangible, practical, and of use to real developers. The MORALE team consists not only of its faculty authors but also its awesome graduate students without whom much of what you will read in this book would only be so many words. They have challenged us with their questions, stimutated us with their ideas, and supported us with their hard work. Finally, we would like to thank the College of Computing at Georgia Tech as a community in the true sense of the word encouraging collaboration, not only with its outward looking aspect, but at a fundamental, architectural level.