

PREFACE

The release of V3R1 of OS/400 saw the introduction of many new development tools: a new version of RPG, the full implementation of the Integrated Language Environment (ILE), database enhancements with triggers, referential integrity (RI), and a host of new and improved application program interfaces (APIs). Having progressed steadily for six or seven years, AS/400 programmers and analysts were suddenly presented with major changes in one fell swoop.

Those who work with RPG are all aware of these new enhancements. All of the trade magazines publish code samples in RPG IV within articles that are full of terms such as *binding* and *service programs*. But how many of us actually use these enhancements? To this day, AS/400 applications implement very few of these features. Programmers can cite a number of reasons for this:

- If it ain't broke, don't fix it.
- We are too busy with Y2K and other projects.
- We don't have the time to learn it all.
- We are waiting for Java.

Or, to put it in a nutshell, it is complex, is of questionable benefit in relation to the time required to learn it, and will be replaced by Java anyway! These common views, as with many common views on the AS/400, are based on misconceptions.

Too Busy

The level of complexity depends on what you want to do; an RPG programmer can grasp the fundamentals of RPG IV within a couple of days. The time invested in learning is not as horrendous as one might imagine and is definitely worth the effort, both directly and in relation to what will happen in the future. Anyone who has programmed within a Windows environment will have few problems understanding ILE. Converting to RPG IV doesn't mean that legacy applications become harder to maintain; in fact, they become easier to maintain. That alone should justify converting to RPG IV.

JAVA

Without a doubt, Java signals a departure for AS/400 shops and will greatly influence the modification of current applications and the development of new ones. It is definitely the wave of the future on the AS/400. Yet, only the most ardent Java zealots would claim that Java will fully replace RPG on the AS/400. The Gartner Group, an independent consulting company specializing in IT trends, projects that by 2003, 40 percent of new AS/400 development will utilize Java. With C accounting for another 20 percent, that leaves about *40 percent* of new development still in RPG, to say nothing of all the legacy applications that will remain in use. No matter how quickly Java develops, it will be a long time before its presence is strong enough to justify rewriting the core of an AS/400 application. RPG is not going away.

After a final comment on Java, I won't mention it again. If you are familiar with RPG IV and ILE, you will find the transition to Java's object-oriented design much easier than if you were to approach it from the traditional RPG III standpoint. An understanding of triggers and referential integrity will show you how DB2/400 easily handles some of Java's apparent shortcomings. An excellent point of reference is *Java for RPG Programmers* by Phil Coulthard and George Farr, published by Advice Press.

HANDLING THE LEGACY

Whatever the reasons and whatever the rights and the wrongs, the result is that many AS/400 programmers have achieved a condition of stasis. We remain stationary while technology moves on and eventually seems light years away. Although many of us think new technology is great, we still rely on the legacy application that runs the business. Is it possible to derive some of the benefits of new technology from within a legacy application? You bet it is. The set of enhancements introduced with V3R1 provides a means of immediately increasing application reliability and programmer productivity with relatively little effort. At the same time, it provides AS/400 programmers with a firm basis for things to come.

Whatever direction the future takes, at the moment the only mistake is to do nothing. While the technology means new concepts to be learned and new ways of doing things, very few ideas are so radically different that they conceivably fall outside the grasp of the average RPG programmer. With all of this in mind, programmers can view this book as a tutorial that shows how to move to RPG IV, ILE, triggers, referential integrity, and APIs by taking a set of sample legacy programs through the process of conversion, re-engineering, and redesign. Let's take a brief look at the chapters.

Chapter 1: The Legacy introduces the legacy application. These are the programs that will be converted, re-engineered, tweaked, and generally abused throughout the book. The chapter explains the standards used in the programs as well as some of the programming peculiarities.

Chapter 2: The Basics of RPG IV examines, as the title states, the fundamentals of RPG IV, including changes to the specification types and the new data specification. A cursory glance at this chapter will suffice for those who have some familiarity with RPG IV.

Chapter 3: The Conversion Process looks at the different conversion options available, including system-supplied and third-party options and using your own. Converting to RPG IV is easy and this chapter shows the different degrees of conversion that can be achieved. This chapter also serves as the starting point for changes we'll make to the legacy code.

Chapter 4: More RPG IV looks at some of the neat features introduced with RPG IV like compiler directives, date and time processing, built-in functions, indicator data structure, prototyped calls, and subprocedures.

Chapter 5: Re-engineering with RPG IV applies to the legacy code some of the features discussed in Chapter 4. This chapter shows how easily you can incorporate these new features with minimal effort.

Chapter 6: The Basics of ILE introduces the concepts of program binding, ILE program structures, and converting to ILE.

Chapter 7: Re-engineering with ILE applies to the legacy code some of the features discussed in Chapter 6. This chapter shows how current programs can be redefined as modules and constructed as new ILE programs that are easier to maintain.

Chapter 8: Triggers looks at the implementation of a trigger program in the legacy application. This chapter shows how the introduction of a trigger program can simplify an existing application and reduce the overhead in future enhancements.

Chapter 9: Referential Integrity shows how the introduction of referential constraints can bolster a design flaw in an existing application and again ease concerns about future enhancements.

Chapter 10: APIs shows how APIs present new ways of approaching traditional problems. User spaces are implemented to help speed up prompt processing in the legacy application.

Chapter 11: The Full Monty takes enhancement of the legacy application to the extreme and introduces a few RPG IV and ILE features not previously dealt with.

The instructional portion of this book closes with a summary in Chapter 12. The appendices, which make up the bulk of the material in the book, provide code listings as the legacy programs go through the different stages of their

metamorphosis. Refer to them often, especially Appendix A, which outlines the legacy application prior to the enhancements we'll make to it.

THE COMPANION CD-ROM

The companion CD-ROM includes the sample code as it goes through the stages of conversion, reengineering, and redesign and a trial copy of Linoma Software's conversion tool, CVTILERPG. Appendix I contains instructions for loading software to the AS/400.

