Preface

A business rules approach to systems development promises to be the most practical and desirable way to build systems. It can help you build better, easily changeable systems faster than any previous approach. This book provides a step-by-step practical approach for building systems using a business rules approach.

The timing of the emergence of a business rules approach is compelling. Adaptability is a subtle but powerful change in the way we build systems. As a parallel, let's look at how the emergence of relational concepts and corresponding technology took place. Ted Codd documented his relational model in a landmark paper in 1970. Subsequently, small software vendors created database management systems based on a subset of the concepts of the relational model. Shortly after that, major software vendors did the same. Around that time, authors like C.J. Date published books and gave presentations that explained the concepts of the new software to the IT industry. Eventually, people such as Candy Fleming and I published books and gave courses in how a practitioner can proceed from gathering requirements to using the new software for creating relational databases.

Relational technology flourished for many reasons:

- It was theoretically sound.
- Software vendors understood the theory, to some extent, and delivered commercial products.
- IT notables, such as C.J. Date, eloquently explained the theory, benefits, and practicality of relational products to the IT industry.
- Other practitioners gained early experience and developed step-by-step approaches for using the new software.

Let's now evaluate the evolution of the business rules approach. Ted Codd laid the groundwork for the integrity aspect of the relational model back in 1970. While this aspect has not been fully deployed by relational DBMS vendors, it had made strides through triggers and stored procedures and in research under the auspices of active database systems.

Innovative software vendors (such as Versata Inc, Brokat Technologies, USoft Inc., ILOG Inc., The Haley Enterprise, and Rule Machines Inc.) incorporated those concepts into a type of software product called commercial rules technology. These products are used mostly by application developers, less so by database professionals, because the product environments promote the idea that those integrity constraints, the business rules, are part of the application development world.

In addition, Ronald G. Ross has published further theory on the classification and grammar of the various types of integrity constraints, or business rules. C.J. Date published a book for IT practitioners that explains the history of the business rule phenomenon as it relates to the relational model, as well as the adoption process for software vendors. He further acknowledges that, while specification of constraints (business rules) can be part of a fully matured relational database management system, there are benefits to doing so in a middle tier that manages the rules independently of the DBMS.

So, as for the business rules approach:

- It is theoretically sound.
- Vendors understand the theory, to some extent, and are delivering commercial products.

- C.J. Date and Ronald G. Ross have explained the theory and product capabilities to the IT Industry.

It is now time for a step-by-step approach for applying the theory to technology in the world of e-business, Web-based, object-oriented, and relational database systems development. This book combines best practice systems design principles plus adherence to the principles of a business rules approach, thereby enriching what you already know how to do.

A system built according to the business rules approach has many advantages over other systems. However, given the pressures of e-commerce, the most important advantage is that a business rules system is designed to easily accommodate changes in the business with minimal disruption. Therefore, this book introduces a new emphasis and formalism around capturing, validating, and automating the rules of the business so that the business can easily change those rules as it sees fit.

The methodology in this book builds on familiar methodologies such as structured systems analysis, information engineering, and object-orientation so that readers can easily add a business rules approach to existing practices. The most significant aspect of this book is the introduction of a rule track into a systems development methodology. The steps, guidelines, techniques, and examples of activities in this track guide the reader in understanding what business rules are, why they are important to the business, why it is important not to bury the business rules in code again, and how to analyze and deliver the rules as a valuable, changeable aspect of the resulting system.

Defining the Business Rules Approach

A business rules approach is merely a formal way of managing and automating an organization's business rules so that the business behaves and evolves as its leaders intend. Essentially, a business rules system is an automated system in which you separate the rules, logically, perhaps physically, and share them across data stores, user interfaces, and applications.

You can apply most of the concepts in this book regardless of whether you utilize special rules technology, although use of such technology will deliver the most immediate and long-lasting benefits. As a starting point, during scoping for example, you will pay special attention to the business motivation for your project. Aspects of this motivation may include objectives, goals, strategies, tactics, and policies. These are the fundamental justification for the rules you uncover, deliver, and change during the systems development effort.

During the discovery of requirements, you will seek important decisions behind the target system and dissect these into atomic rules, capturing them in a rules repository. You can extend an existing repository, utilize an existing CASE tool, or build a simple rules database.

Moving into analysis, you become involved in steps and techniques specifically aimed at verifying the quality of rules, connecting rules to information and knowledge, and understanding dependencies among rules.

Finally, in the design phase, you consider implementation options, paying close attention to those that allow you to deliver the rules in a way that accomplishes four significant objectives:

- *Separate* rules from the rest of the system so you can reuse them
- *Trace* rules to the reasons they exist as well as to where they are implemented
- *Externalize* rules so everyone can know what they are