Foreword

AS I WRITE THIS FOREWORD in July 2006, I know something big is about to happen.

Developers are discovering the power of having machine-readable representations of their programs that match their intention. This idea is sometimes referred to using the shorthand phrase *code as data*.

More importantly, developers are realizing that no runtime architect or language designer is all that qualified to define the schema for that data—a domain expert is the only one who has the right expertise. This idea is sometimes referred to as *domain-specific languages*.

Systems like Smalltalk, the Java Virtual Machine (JVM), and the Common Language Runtime (CLR) have proven that there is value in having machine-readable representations of type definitions for things such as reflection, serialization, and generative programming. However, the basic representation of a type (e.g., fields, methods, classes) is largely a closed world, which doesn't allow users to model (as data) things such as control flow, concurrency, logic constructs, or domain-specific ideas such as discount policy or dotted whole note except as largely opaque instructions strewn across multiple method bodies.

People are now asking themselves, "If my type definition is available as data, why aren't other design-time constructs as well?"

Fortunately, one of the people asking those questions back in 2003 was my good friend Dharma Shukla, who was then working on the Biztalk Server team.

When the Biztalkers set out to generalize their XLANG orchestration engine as an embeddable platform component, they could have simply taken the language constructs from XLANG and put yet another XML syntax around them (that proposal was certainly on the table). This is the most obvious approach and would have satisfied their charter for building the Windows Orchestration Engine (WinOE) perfectly well.

Fortunately, Dharma had the foresight to know that he wasn't the right guy to define "the one true schema" for all programs and instead decided to "go meta" and steer the project toward building an extensible runtime that allows users to define their own opcodes that match the domain they're working in. Add to this the decision to support defining and composing these opcodes in an XML dialect and you have a system that lets anyone decide what the vocabulary and sentence structure is for describing applications in a given domain.

As Dharma Shukla and Bob Schmidt so aptly show with this book, Windows Workflow Foundation (WF) is an excellent example of a meta-runtime that puts developers in control of how programs are written and how programs execute. The developer defines the schema for a program and then provides an interpretation over that schema to allow it to be deployed and run. It is a simple idea that has huge ramifications.

Like I said, something big is about to happen.

Don Box
July 2006, Yarrow Point, WA