IBM Rational Business Developer Extension: An executive overview

Level: Introductory
Stefano Sergi, Product Line Manager, Rational® Business Developer and EGL, IBM

15 May 2007

The importance of information systems in improving businesses efficiency and competitiveness continues to grow, particularly in a global economy characterized by constant search for cost optimization and penetration into new markets with mergers, acquisitions, and requirements for out-sourcing or right-sourcing. The explosion of computer networks and the global reach of the internet are transforming the way businesses operate and use the information. The era of traditional application islands is rapidly coming to an end.

New integrated information solutions for improving the way businesses enable partners, customers, and employees are at the top of many CIOs' agendas. But to avoid the pitfalls of application silos which hinder responsiveness and flexibility, the enlightened CIO is embracing the services model as the architectural underpinning of all new business software.

This new generation of IT systems requires software development capabilities that support new middleware and emerging application architectures. At the same time, however, development teams have grown accustomed to levels of productivity and simplicity that accompanied the emergence of client/server computing. IBM has evolved the capabilities of the IBM Rational Software Delivery Platform to address this need with an integrated set of tools, methodologies, and best practices that enable effective governance of the software development process and accelerate the transition to SOA.

The new product is called IBM Rational Business Developer Extension. It equips client/server based operations teams, especially those working in COBOL or other heritage language environments, with the tools to deliver services rapidly without imposing the bottlenecks of extensive training in Java. Rational Business Developer Extension (RBDe) provides development teams with a language IBM introduced in 2006, Enterprise Generation Language (EGL). This article describes the reasons IBM developed EGL and discusses how IT organizations can use it to become more productive, more quickly in today's services-based arena.

Why EGL? A historical perspective

At the heart of the IBM Rational Software Delivery Platform are a set of SOA design and construction capabilities provided on top of the Eclipse workbench. These include powerful and highly productive tools to organize and speed up the work of software architects, designers, Web developers, and Java developers. However, unlike the majority of the other leading tool vendors in the industry, IBM recognizes that the strength and popularity of Java -- with its countless systems and middleware APIs and productivity boosting frameworks -- also poses significant technical challenges that require a long and continuous learning process.

This issue is not new in our industry. Each computing technology revolution has presented new technical programming challenges that tax developers' time in terms of learning and coding new interfaces at the expense of coding business functionality. Here are a few examples. With the advent of online transaction processing (OLTP) in the early 1970s, application developers had to learn and master the services interfaces (typically COBOL or Assembler APIs) of OLTP

middleware such as CICS and IMS. And with the advent of relational data, graphical interfaces, and client/server computing, developers had to learn and code the intricate interfaces of database middleware and GUI services middleware. Still later, with the explosion of application integration, developers had to master a whole new set of skills relative to using Application Integration Middleware (AIM) layers, from message-oriented middleware, to adapters or broker APIs.

While in some cases it is essential to have complete access to the full systems interface capability, application developers can address a significant portion of requirements through typical usage patterns abstracted in a way that provides a simpler interface and shields the programmer from low-level coding.

To continue the historical perspective, think of how capturing usage patterns of GUI widgets or relational data tables into visual tools and language integration with such tools brought significant levels of productivity into the creation of many desktop and client/server applications. That was the era of PowerBuilder, Visual Basic, and Delphi. Accomplishing more sophisticated or specialized requirements still required writing C or C++ programs, but the vast majority of the applications could be written using these simpler and more abstract programming environments that were easier and quicker to learn and, in general, made programming teams more productive.

In today's era of application servers, application integration, and SOA, the developer community faces a similar predicament. This is why IBM Rational has introduced the Enterprise Generation Language (EGL) as an integral part of the Eclipse-based programming workbench of the IBM Rational Software Delivery Platform. A state-of-the-art integrated development environment (IDE) for EGL is available in the Rational Business Developer Extension v7.0.

Compared to Java and the Eclipse environment, EGL is easier to learn, and makes development teams more productive, thus complementing the breadth and depth of Java with a simplified and more abstract development paradigm. Developers with practically any skills background can quickly deliver cross-platform, transactional data-centric services and applications.

EGL is not intended as an alternative to Java, but rather as a way for IT organizations to embrace and leverage the strengths of Java EE without forcing the entire team of developers to learn and master the lower level intricacies and complexities of Java programming. Much of, or even all of, an application might be delivered solely using EGL; although as with PowerBuilder and other 4GL tools of yesterday, there may be instances where programming in Java may be required or even appropriate.

EGL is squarely targeted to the business oriented development teams, who value ease of learning and high productivity, who need to deliver modern applications and services but cannot afford the time, cost, and risk to transform each developer into a Java expert.

What is EGL?

Enterprise Generation Language (EGL) is a modern programming language specifically designed to help the business oriented developer leverage all the benefits of the Java platform while avoiding learning all of its details and focusing on the business application domain. EGL's simplified high-level specifications let you quickly write full-function applications and services based on Java and modern Web technologies. Developers write their business logic in EGL source code using the powerful development facilities of Rational Business Developer Extension (RBDe). From there, the RBDe tools generate Java or COBOL code, along with all the runtime artifacts you need, to deploy the application to the desired execution platform.

EGL hides the details of the Java platform and any associated middleware programming interfaces. This frees developers to focus on the business problem rather than underlying implementation details. Developers who have little or no experience with Java or Web technologies can use EGL to create enterprise-class services and applications quickly and easily.

EGL does not provide low-level APIs to operating systems and subsystems; therefore, while it is not suited for systems development, it is particularly powerful for transactional business services and applications development. Full interoperability with Java and tight integration with the Eclipse Java IDE adds the required flexibility to fulfill the broadest and most comprehensive spectrum of systems requirements.

What applications can be built with Rational Business Developer Extension?

Rational Business Developer Extension is uniquely designed to address the full spectrum of business application requirements. These include:

- **Business services**. The language's built-in notion of Service allows to create and consume Services in an extremely simple and straightforward way. The availability of this programming construct drives systems implementation to adhere to the sound architectural principles of service orientation.
- **Web applications**. Tight integration of EGL with Java Server Faces framework and tooling allows creation of Web applications in a simple and productive way, without the need to know neither java nor the details of the JSF framework.
- **Reports**. EGL integration with the Jasper Reports open source reporting engine allows creation of professional reports.
- **Batch systems**. The language includes the notion of batch program, which can be generated to run without end user interaction, for example for reports production or for batch database load/update.
- **Text UI applications**. In order to facilitate migration of legacy systems to a modern development environment, EGL includes the ability to create traditional character based (green screen) user interfaces and relative programming constructs. This capability can also be used for new development if needed.
- **GUI applications**. EGL's latest enhancements provide the ability to create applications that interact with end users through a graphical user interface. The initial support for GUI applications enables Informix users to migrate their current i4gl applications that use the Console UI interface, but IBM plans to deliver further rich UI support in the next release of the Rational Business Developer extension.

How EGL applications target execution

The Rational Business Developer Extension comes with a code generation engine that transforms EGL source for application and services into either Java or COBOL source.

Generated COBOL can be deployed to the following environments:

- z/OS v1.6 or later: batch, IMS or CICS applications
    - IMS/ESA V7.1 or later (for IMS applications)
    - CICS TS v2.2 or later (for CICS applications)
- iSeries v5.3 or later

Generated Java can be deployed to the following environments:

- z/OS v1.6 or later
    - IBM SDK for z/OS Java 2 Technology Edition v1.4, for execution of Java programs under USS
    - WebSphere Application Server for z/OS V5 or later for execution of Web applications

- iSeries v5.3 or later
    - JDK v1.4 or later, for execution of non J/EE Java programs
    - WebSphere Application Server v5.1.2 or later for execution of Web applications

- AIX v5.2 or later
    - JDK v1.4 or later, for execution of non J/EE Java programs
    - WebSphere Application Server v5.1.2 or later for execution of Web applications
    - Apache Tomcat 5.5 or later for execution of Web applications
- Windows 2000, 2003, XP
    - JDK v1.4 or later, for execution of non J/EE Java programs
    - WebSphere Application Server v5.1.2 or later for execution of Web applications
    - Apache Tomcat 5.5 or later for execution of Web applications
- Linux (RedHat 3 or higher, Suse 9)
    - JDK v1.4 or later, for execution of non J/EE Java programs
    - WebSphere Application Server v5.1.2 or later for execution of Web applications
    - Apache Tomcat 5.5 or later for execution of Web applications
- HP Unix 11iv2
    - JDK v1.4 or later, for execution of non J/EE Java programs
    - WebSphere Application Server v5.1.2 or later for execution of Web applications
    - Apache Tomcat 5.5 or later for execution of Web applications
- Solaris 9 or later
    - JDK v1.4 or later, for execution of non J/EE Java programs
    - WebSphere Application Server v5.1.2 or later for execution of Web applications
    - Apache Tomcat 5.5 or later for execution of Web applications

Deployment of EGL Java applications requires the installation of a set of runtime libraries (JAR files) in the target environment: these are shipped free of charge as part of the Rational Business Developer Extension v7.0 product.

Deployment of EGL COBOL applications for iSeries requires the installation of a set of runtime libraries in the target environment: : these are shipped free of charge as part of the Rational Business Developer Extension v7.0 product.

Deployment of EGL COBOL applications for zSeries requires the installation of a set of runtime libraries in the target environment which are available in the IBM Rational COBOL Runtime for zSeries V6 product

Supported databases

EGL provides simple and powerful language elements that accelerate access to data stored in a variety of databases. Programmers do not need to be proficient in SQL data manipulation language, JDBC, or other SQL access programming technologies, nor do they need to learn file or hierarchical databases access APIs. Using a small set of polymorphic verbs, they can write full function services accessing the following data stores:

- DB2 7 and later
- IDS 7.3 and later
- Oracle 8 and later
- SQL Server 2000
- Derby 10
- VSAM
- IMS/DB (DL/I)

What makes EGL powerful?

EGL empowers the business-oriented developer to be extremely productive in a very short time based on the SOA-related capabilities shown in Figure 1.

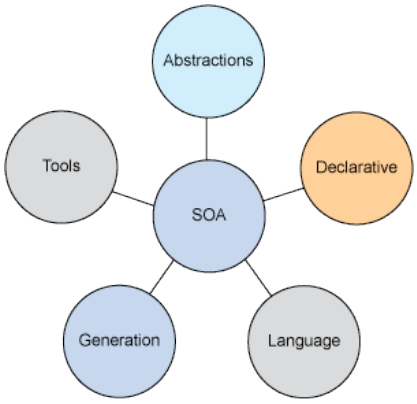

Figure 1: EGL supports a variety of SOA-related capabilities that help business-oriented developers become extremely productive in a short time.

- **Abstraction**. EGL provides concise and powerful notations that eliminate tight coupling and reduce the amount of coding required to interface systems and middleware. This abstraction significantly simplifies and speeds up the developer's work.
- **Declarative programming**. EGL includes, where possible and appropriate, a certain level of declarative specifications aimed at reducing repetitive and error-prone coding (for example, by associating a validation rule to a data item, every time the item is used in a certain context the validation is automatically applied and enforced.)
- **Language**. EGL provides a comprehensive but easy to learn language notation that is modern and modular, with a rich library of built-in functions to boost developer productivity for commonly required operations (such as date and time math, string manipulation, and so forth). Additionally, the language is extensible and offers full interoperability with other languages. In particular, it provides EGL interfaces to native Java.
- **Generation**. Though simplified, the EGL development technology must still guarantee optimal deployment to the runtime platforms to take advantage of their Qualities of Service (QoS) and to allow native management and monitoring of the systems in operation. This is accomplished through a code generation engine available as part of the Rational Business Developer Extension that transforms the EGL specification into native Java or COBOL source, and creates any other required deployment artifacts.
- **Tools**. To further boost developers' productivity, Rational Business Developer Extension provides a rich set of tools built upon the Eclipse IDE framework. These include EGL source animation for debugging purpose, powerful smart editing, visual construction, graphical navigation, and specialized tools such as tight integration of EGL notations with graphical Web development tooling, automatic transformation of UML models or database schemas into completely functional EGL services and applications.
- **SOA**. As mentioned in the introduction to this document, service orientation is becoming a critical requirement for delivering new business solutions; therefore EGL has been designed from the ground up with this requirement in mind and a simplified and abstracted SOA development paradigm has been built into the language itself and complemented with tools and generation, consistently with the basic tenets of the power of EGL discussed above. The availability of the "Service" programming construct drives systems implementation to adhere to the sound architectural principles of service orientation. Developers of any skill can easily create services without the need to know Web Service protocols and standards such as WSDL, SOAL, XML etc.

The value of EGL and Rational Business Developer Extension

Organizations that need to deliver modern, powerful Web and SOA solutions can enjoy the following benefits:

- Increased programmer productivity
- A newly empower class of developers delivering modern solutions and SOA
- Increased flexibility and responsiveness by eliminating skills silos
- Reduced training costs
- Reduced errors and deliver higher quality
- Simplified legacy integration
- Reduced development and maintenance costs via a flexible choice of deployment options, including:
    - All major IBM Operating Systems and runtimes
    - Cross platform solutions

Who benefits from using Rational Business Developer Extension?

EGL is an excellent solution for any application development organization that is either 1) starting new projects that require Web development or SOA and are faced by skills challenges and tight deadlines, or 2) planning to modernize and improve their current software development practices and tools. Let's consider the various platforms, skill sets, and related conditions for which Rational Business Developer Extension was designed.

Organizations using System i and System z
In-house development teams typically fall into two fairly distinct camps: the legacy developers working with RPG, COBOL, 4GLs, etc., and the "new" developers working with GUIs, Windows, Java, the Web, etc. If legacy development is large and there is mounting pressure to deliver new solutions to the business, the EGL value proposition will be attractive.

Another situation that EGL can improve is when separate "islands" of developer teams prevent optimal use of the programming resources. When legacy developers cannot easily do Web or SOA programming, and "new" developers are unable to deal with legacy code, then managing cross-technology teams is challenging and slows things down. EGL can be learned easily by either team, allowing a unified pool of developers to work on end-to-end projects via a single technology platform.

Another important value EGL can bring to these organizations is the motivation of developers through a modern powerful programming environment designed with the business developer in mind, which can help retain existing talent and attract new talent.

Organizations with obsolete 4GL (IBM and non-IBM)
Teams that currently use or have used a Fourth Generation Language (4GL) may be interested in EGL for a number of reasons. First, these teams typically value high productivity programming, or they would not have invested in a 4GL in the first place. If the 4GL is no longer capable of delivering value in the transformation to Web and SOA (either because the vendor has stabilized the products or because the requirements of the new systems exceed the 4GL capabilities), then EGL can be a good vehicle for extending the life of the investment by leveraging the skills of the 4GL developers.

EGL can be quickly learned by 4GL developers, and the migration/transformation tools and services can repurpose the old 4GL code into EGL, breathing new life into old but still valuable code.

The following IBM 4GLs are easy to migrate to EGL using tools included with the EGL development environment: **Informix 4GL, VisualAge Generator, CSP, VisualGen**.

The following non-IBM 4GLs can be transformed into EGL using third party tools and services: **Natural, CA (COOL-Gen, COOL-Enterprise, Ideal), HPSeer**.

Organizations with developers new to Web applications or SOA using Java and J/EE
Organizations who are planning to develop Web and/or SOA applications but have limited or no experience in the Java and J/EE platform, or have developers with limited or no object-oriented development expertise, cannot afford the time and costs of re-training their staff. And typically they are adverse to contracting out the development to third parties because they want to avoid long term dependency and costs. These organizations should take a close look at EGL.

Regional and Global Systems Integrators
EGL can be particularly valuable to this type of organization. Systems Integrators who build custom business solutions for other companies will increase their competitiveness if they can reduce development costs, train new developers more quickly, and count on a pool of developers who are flexible and not locked into a "single target platform" skill.

Moreover, Systems Integrators are often involved in large rewriting or modernization projects, and EGL can become a significant advantage in their bids, especially when used in combination with automated transformation tools. The automated tools mentioned earlier also provide the ability to transform aging COBOL applications into EGL.

Independent Software Vendors
EGL can present a tremendous competitive advantage for ISVs and serve as the source for new market opportunities. Unlike Systems Integrators, ISVs create general purpose business solutions which they resell, along with customization services, to companies who prefer off-the-shelf solutions to custom solutions. One particular advantage is the flexibility in platforms targeted for their solution, and of course EGL can help reduce development costs and training schedules, allowing for a more flexible workforce

Common questions

Here are some common questions, and objections, we have encountered, the answers to which I think you will find valuable.

Q: Is performance of the generated application adequate?

A: Applications generated with Rational Business Developer Extension have been benchmarked against identical hand-written applications and have shown comparable levels of efficiency and performance. Although hand-crafted code can be tweaked to achieve superior performance, in general for typical business applications you should see no perceptible difference.

Q: I have never heard of EGL, so it can't be as popular as other languages.

A: IBM introduced this new technology in the spring of 2006, and we are seeing a tremendous interest. While the latest support for Web standards like JSF and for Web Services is totally new, and a significant portion of the EGL technology is based on very popular and widespread IBM tools, such as VisualAge, which are used by thousands of customers worldwide.

Q: Why would I want to adopt EGL, which is a proprietary IBM language?

A: EGL is the continuation of more than twenty-five years of IBM's investment in research and development in rapid development technologies. With the emergence of new computing models, such as Web and SOA, IBM views a simplified programming approach as critical to organizational success and has therefore invested significantly in this strategic solution and is fully behind it. EGL is not only the way forward for many strategic IBM accounts who have significant investment in IBM rapid development technology, but is positioned as the strategic, rapid development language with plans for industry standardization.

Q: Where do I find skilled EGL developers?

A: There is an extensive network of consultants and IBM business partners whose expertise can help you get your projects off the ground quickly. Your in-house developers can ramp up the required skills in a matter of weeks, eliminating the need to compete for hiring scarce skilled resources.

Q: Code generators are good for productivity, but if something goes wrong in production, my developers will not be able to do problem determination because they will not understand the generated code.

A: The EGL generator allows a developer to easily track back from generated code to the original EGL source, and to run debugging tools that step through the symbolic source while the generated code is executed.

Q: Code generators may be easier and quicker than hand-coding in Java, but they always require a special runtime which makes it hard to achieve the security, scalability and reliability offered by native servers.

A: Applications generated with EGL can deploy as native J2EE, CICS, or IMS programs, exactly as you would deploy hand-written applications. The runtime libraries used by the generated code are simply packaged as commonly used, auxiliary shared code that would not make sense to repeat as generated code in each and every program. This is a good software engineering practice that most customers already use in their own application architectures.

Competitive positioning

The application development tools market is extremely fragmented and undergoing a profound transformation with the emergence of two major trends: 1) the open source movement, and 2) the network computing and SOA movement. The market is filled with innumerable programming languages and relative Integrated Development Environments (IDE), each promising to be the most productive and easiest to use, and each filled with a wealth of tools, wizards, and programmer shortcuts and aids (smart editors, visual designers, powerful browsers and project organizers, interactive debuggers, code templates or auto-coding aids, code generation wizards, etc). Rational Business Developer Extension is second to none in IDE power, and takes full advantage of the mature and state-of-the-art Eclipse open source tooling framework.

While a detailed comparison of IDE features is beyond the scope of this discussion -- given that a fair positioning and comparison of technologies should be based on a comprehensive set of criteria -- IBM asserts that a close examination of the most critical IDE requirements reveals the superiority of EGL.

Most organizations will agree that the following requirements are critically important to support their application development objectives:

- **Versatility**. Ability to support different application requirements (e.g., types of applications you can build such as desktop, Web, batch, SOA, etc.), different target platforms (e.g., operating systems) different middleware (e.g., databases, MOM, transaction mgmt servers, etc.)
- **Ability to deploy natively**. The executables can be deployed without need for special proprietary servers to leading scalable robust enterprise transactional environments (e.g., J/EE servers, CICS, etc.)
- **Legacy support**. Easily integrate and use existing applications, and/or to provide a way to transform and modernize aging but valuable business systems.
- **Openness (runtime)**. Create applications that inter-operate with other systems through support for open standards and participate in a SOA.
- **Openness (development)**. The extensibility of the language and of the development workbench

- **Language**. Two key aspects, the level of abstraction and productivity offered to the developer, as well as the computational completeness and robustness of the language.
- **Ease of learning**. How quickly developers can be fully productive
- **Lifecycle integration**. How well the technology integrates within a robust lifecycle solution

A technology such as EGL is attractive for its fundamental value proposition -- easy to learn, easy to use, productive, robust, enterprise class application/services development technology for business oriented developers. Any tool based on complex low-level languages such as C#, C++ or Java, however powerful, should not be considered a competitor for EGL.

When evaluating EGL you should consider how alternative tools and technologies measure based on the list of characteristics above. We are confident you will find EGL to measure up very well in all those areas against any major competing solution.