A technical discussion of the IBM Software Development Platform
White paper

Realizing the IBM Software Development Platform

By Alan W. Brown
IBM Distinguished Engineer

April 2004

Contents

Introduction

The IBM Software Development Platform is an evolving vision for software development that recognizes the important role played by software in the fabric of on demand organizations. It highlights a new perspective on the importance of software development as a key business process for organizations that rely on software to run their businesses, or that include a significant software component in the goods they manufacture, distribute or sell.

However, the IBM Software Development Platform is much more than an interesting concept. There is practical value in the IBM Software Development Platform to today's on demand enterprise. In this whitepaper, we focus on what the IBM Software Development Platform means in the context of three key customer goals:

- *Enabling business integration and transformation*
- *Managing software development assets throughout the software life cycle*
- *Adopting an effective, open technology infrastructure*

By focusing on these three goals, we introduce the solutions available from IBM Software Group today, explore the technological underpinnings of the IBM Software Development Platform and highlight the planned evolution of those solutions as they leverage that technology.

The role of IT in business

Over the past decade, there has been growing recognition of the important role of information technology (IT) in driving today's Internet-based economy. Worldwide spending on IT systems was estimated to be around US$2.4 trillion in 2001.[1] Software has been the linchpin for many organizations as new markets were created, major new corporations grew at lightening speed, new business alliances and supply chains were formed, and traditional markets were transformed. Of course, the euphoria of the 1990s has now given way to the business realities of the new millennium. This has been characterized as a return to traditional business economics, a "back-to-basics" movement that has reminded many organizations that success is built on high-quality products that address high-priority customer needs and provide a measurable return on investment (ROI).

The focus on business basics and economic realities has led a number of people to reexamine the role that information systems play in business, and the ROI of IT investments. Commenting in a recent *InformationWeek* article,[2] General Motors CIO Ralph Szygenda summarized the concerns driving IT by saying:

> *… business-process improvement, competitive advantage, optimization, and business success do matter and they aren't commodities. To facilitate these business changes, IT can be considered a differentiator or a necessary evil. But today, it's a must in a real-time corporation . . . I also agree on spending the minimum on IT to reach desired business results. Precision investment on core infrastructure and process-differentiation IT systems is called for in today's intensely cost-conscious business versus the shotgun approach sometimes used in the past.*

Szygenda's comments highlight that IT investment is important, but also that those investments must be targeted at the business processes that help to differentiate an organization, drive business change and control costs. Hence, successful management of business priorities requires that organizations take a new perspective on the role that software plays in achieving the organization's business goals.

Software as a business process

Business transformation occurs by integrating and automating horizontal business processes. Initial efforts at business transformation focused on common major infrastructural concerns, such as enterprise resource planning (ERP), supply chain management (SCM), human resource management (HRM) and customer relationship management (CRM). Today, key elements of these business processes can be purchased off-the-shelf, providing standard ways to execute these functions, and reducing risk in their deployment, operation and evolution.

However, there are many other business processes that are unique to each business. Integrating and automating them cannot be achieved solely by purchasing packaged applications. Customization is required in order to capture business rules and strategies that embody key practices that differentiate a company from its competitors. Examples of these processes for specific industries may include:

- *Insurance industry—underwriting, customer rating and claims processing.*
- *Financial services industry—trading and brokering services, portfolio management and settlement actions.*
- *Travel and transportation industry—freight management, asset maintenance and equipment utilization.*

Because this integration and automation is unique to each business, it is also the key to achieving strategic, competitive advantage. Successful organizations not only automate business processes, they achieve integration across these business processes, monitor them in execution and provide real-time feedback to improve business processes in light of changes to their customers' needs.

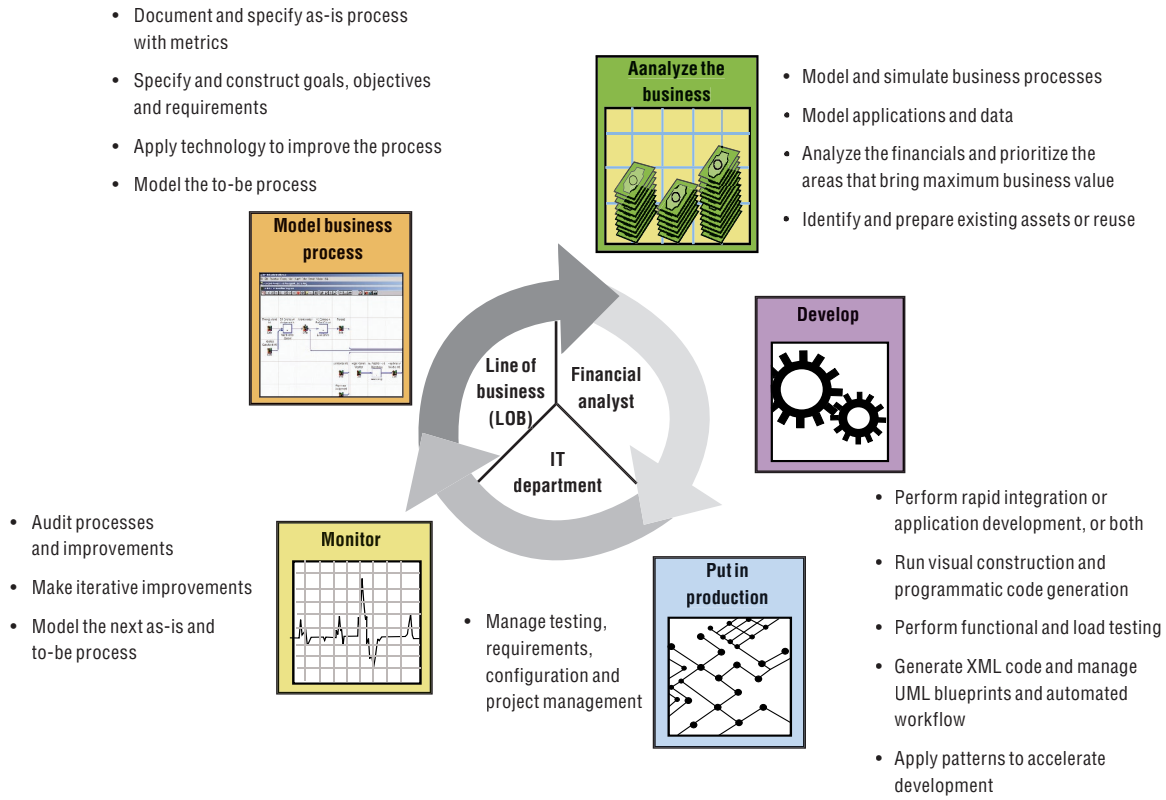

Figure 1. The role of IT in the model-driven enterprise.

Software is key to achieving business transformation. There are five major actions that organizations take to automate and integrate their strategic business processes:

- *Build new applications*
- *Modernize existing applications*
- *Extend packages and existing applications*
- *Integrate new, existing and packaged applications*
- *Deploy new, existing and packaged applications*

Not only is software development key to integrating and automating other business processes, but it is a strategic business process in itself. As such, the process of software development benefits from the same type of horizontal integration we apply to SCM, CRM and HRM.

When we compare software development with other business processes, we see that each is composed of activities that were once considered to be separate and distinct, and supported by different IT systems. Over time, business integration has evolved to the point where today, these activities are recognized as components of a single, horizontally integrated business process supported by a single integrated application: HRM, CRM, SCM and so on (see Figure 2)..

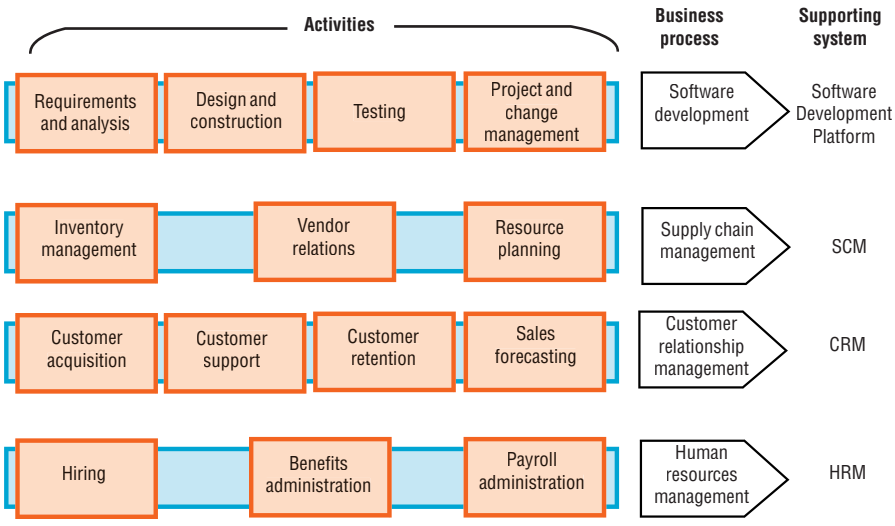

Figure 2. Software development: A strategic business process.

Software development is at the beginning of this transformation, with companies increasingly realizing the value of an integrated software development platform in improving the efficiency of interrelated software development activities. Viewing software as a business process is important because it highlights three major concerns:

- *Connecting business needs with IT solutions*
- *Enabling teams of practitioners*
- *End-to-end visibility, cost containment and risk management*

With the introduction of IBM Rational® Suite in 1999, IBM was the first to provide this integrated software development solution to the business community.

What that means to IT-driven businesses

After this discussion the role and importance of a software development platform, it is useful to briefly examine an example of how the IBM Software Development Platform is used in practice. In this section, we provide an example of the IBM Software Development Platform in use, drawn from a real-world customer scenario.[3]

Example: Connecting business and IT

One of the primary challenges to be addressed in developing enterprise-scale solutions is to connect the domain-specific requirements expressed by business analysts with the technology-specific solutions designed by IT architects. Typically, the connection between these two disparate worlds is very low bandwidth – the two communities have very different skills, use different modeling concepts and notations (if at all) and rarely understand the mapping between those concepts. The IBM Software Development Platform is intended to assist with this problem. In particular, the integration of process, assets and deliverables is aimed at connecting these two different aspects to the system in a precise, automated way.

Let us consider how we would approach the problem of redesigning an airline's flight planning system. The process begins by modeling the business process in an intuitive, easy-to-use notation accessible to business analysts, as illustrated in Figure 3.

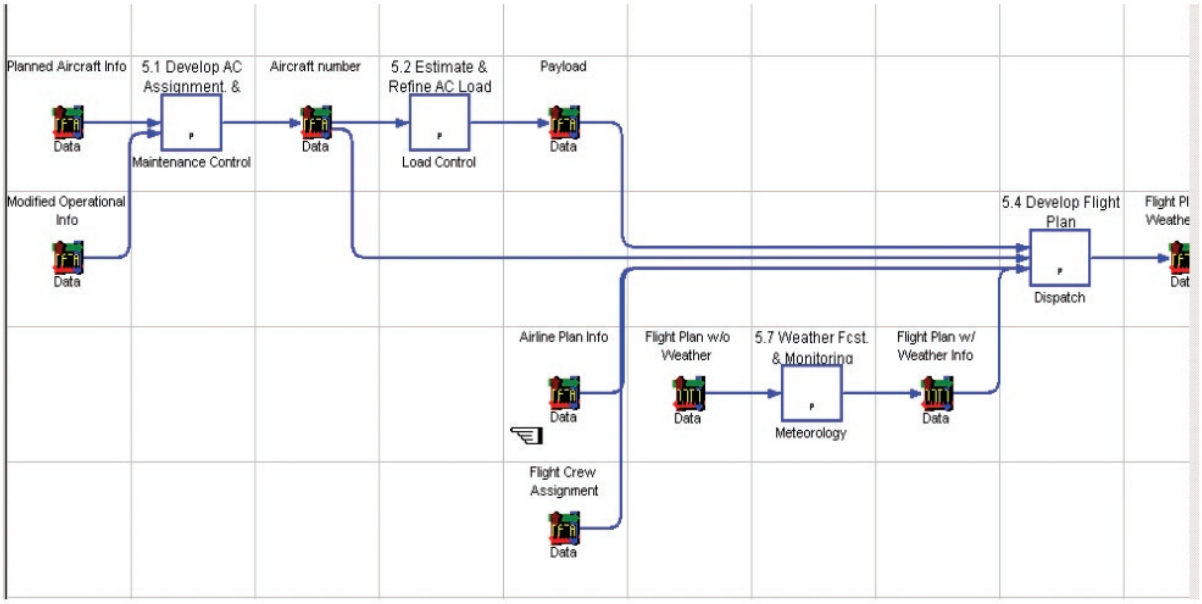

Figure 3. An example of the business process model.

As shown in Figure 3, a business process model captures the key business activities and workflows, in this case using the IBM WebSphere® Business Integration Modeler product. This allows the current system of automated and manual steps to be understood, and potential changes to the system to be designed, simulated and costed before the organization commits to any changes to the business process.

After it has been completed, decisions can be made about which pieces of the new business process should be automated in the software. These can be automatically transformed into an initial set of use cases for the proposed system. in this example, the activities are automatically transformed into use cases in IBM Rational Rose® XDE™ software. The mapping between business activities and use cases can initially be quite straightforward, and then the resulting use cases can be elaborated to add further detail, as illustrated in Figure 4. This realizes the mapping from the domain-specific concepts of the business analyst into technology-specific concepts of the IT architect.

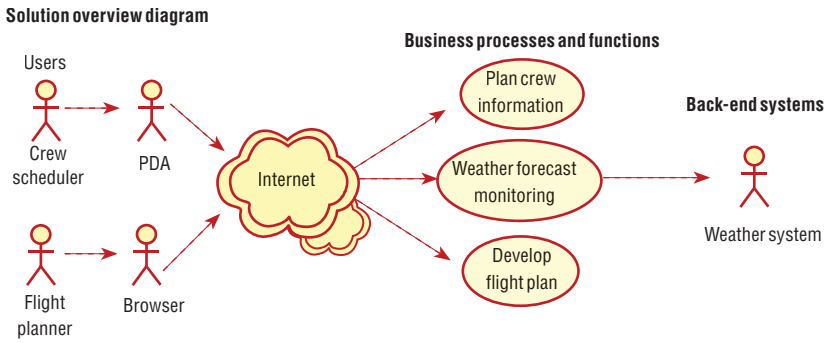

Figure 4. Use case model.

At this point, an initial high-level architecture of a solution should be proposed and refined. A number of patterns are available to guide the architect in choosing an architecture. In this example, the IBM Patterns for e-Business provide one set of architectural solutions that have proved to be useful in practice. The architect selects one of these patterns that matches the particular characteristics of his or her problem domain and binds the various variability points in the pattern to previously defined model elements. The result, as illustrated in Figure 5, provides the initial solution blueprint. In this example, the transformation is realized by applying a predefined IBM e-business pattern in the IBM Rational Rose XDE product.

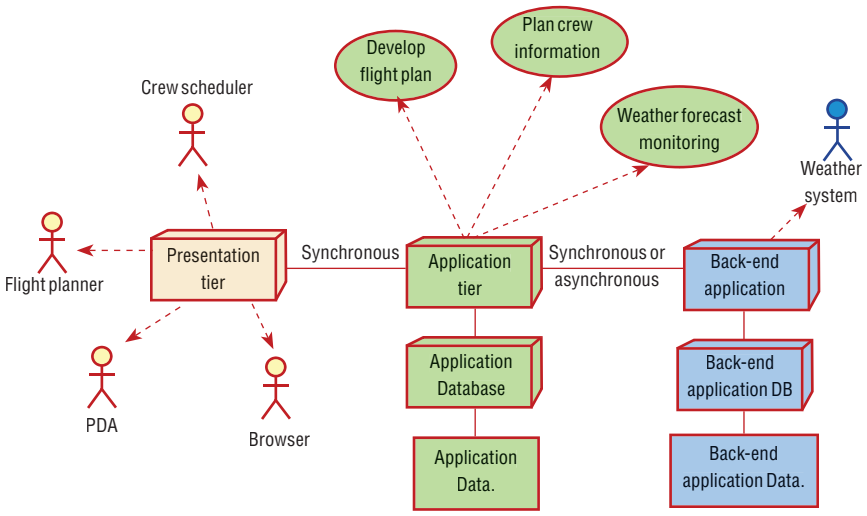

Figure 5. An architectural model.

Patterns such as the IBM Patterns for e-Business provide strategies for refining the initial solution through the application of further patterns that transform abstract model elements into more-concrete model elements. Eventually, this results in the application of a set of deployment patterns that realize the mapping of the solution to a specific physical topology, as illustrated in Figure 6. This completes the mapping into the underlying technologies of choice.

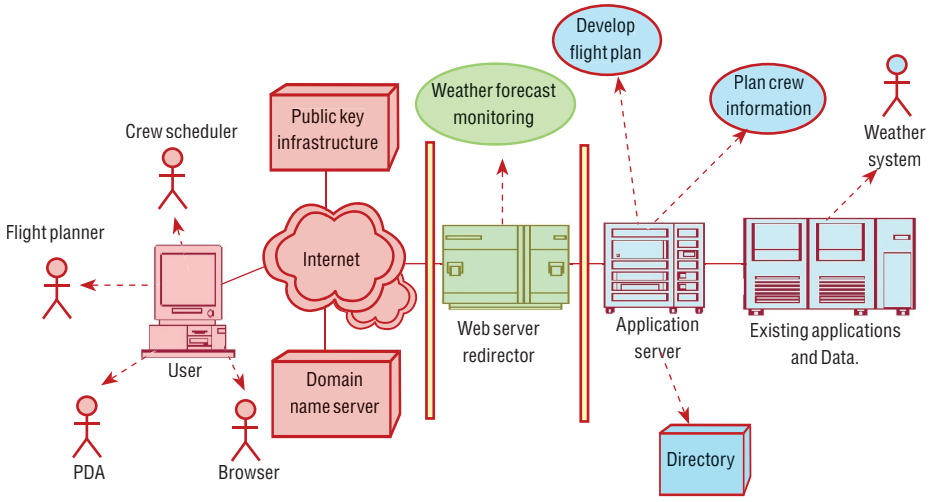

Figure 6. A deployment model.

Further refinements would then take place to add details of the particular platform products that had been chosen and the physical characteristics of those products (for example, the particular application server, messaging infrastructure and database management system).

Additionally, the behavior of the system must be realized by connecting to existing services, developing new functionality, and so on. Here, traditional Unified Modeling Language (UML) modeling approaches can be applied, such as class modeling for describing the data being manipulated, behavioral modeling to describe the business logic and workflow and user interface modeling to define the user interactions. Here again, patterns are applied to transform these models, with the final step being patterns expressed as code templates that generate code from model elements based on a predefined set of transformation rules.

An important result of this real-world example is that the client organization was able to gain visibility in the process of connecting domain-specific business needs into a technology-specific solution following a repeatable, predictable process. The lessons of this example are now being applied in a family of similar solutions in the transportation domain.

Summary: Driving business change with the IBM Software Development Platform

To manage costs and improve predictability, investments in IT must be closely related to the business goals of an organization. Unfortunately, too often, the business drivers are expressed in ways that cannot easily be related to the IT systems, and changes in IT systems are not related back to have an impact on the business. Here, we highlighted an example of a business-driven approach to IT that illustrates how business activities can very explicitly provide the context for software development. This approach applies best practice solutions to improve the predictability and reduce risk in software development, implementing the business-driven life cycle.

The technical underpinnings of the IBM Software Development Platform

As the earlier example illustrated, there are tangible benefits that can be realized today with the IBM Software Development Platform. The integrated set of capabilities offered today supports a business-driven approach to software development across the software life cycle. Many customers in a range of industries are benefiting from this approach today.

IBM is committed to evolve the IBM Software Development Platform to offer users a richer experience for business-driven solutions. In particular, IBM investments are targeted at achieving greater openness for customization and extension by customers and partners, enhanced integration for synchronization of artifacts and assets, real-time feedback and monitoring of tasks and artifacts through improved management and visualization, and coordination of teams across the software life cycle.

As illustrated in Figure 7, the proposed next-generation IBM Software Development Platform will be characterized by improvements in a number of areas, particularly with regard to its role-based focus and its improved integration across the software development life cycle.

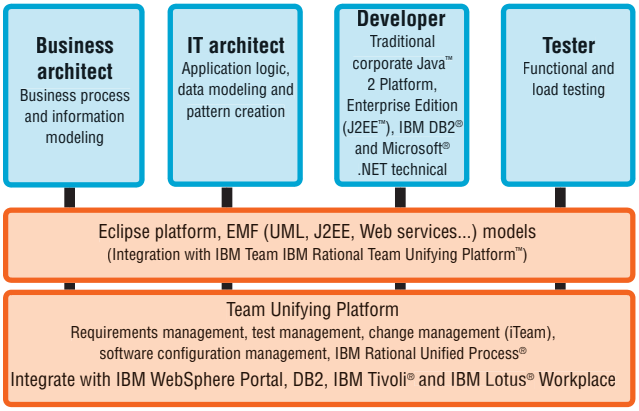

Figure 7. Desktop tools integration direction.

To achieve this, the IBM Software Development Platform will be based on a sophisticated technical infrastructure that consists of five key elements:

- *Eclipse*
- *The Eclipse Modeling Framework (EMF)*
- *The open Model-Driven Development (MDD) platform*
- *The WebSphere programming model*
- *A role-based solutions portfolio*

Eclipse

Eclipse is an open source development project aimed at providing a highly integrated tool framework. The main components of Eclipse include a generic framework for tool integration, and a Java development environment built using this framework. Additionally, many other projects extend the framework and have built tools using the framework to support specific kinds of development approaches and technologies.

At the heart of Eclipse is an extensive tool framework offering a set of core capabilities that supports extension through a plug-in architecture. When companies build solutions based on Eclipse, they are most often creating a set of plug-ins that extends and customizes the Eclipse platform. Furthermore, the architecture of Eclipse itself is a framework and set of tools that are both composed of plug-ins (see Figure 8).

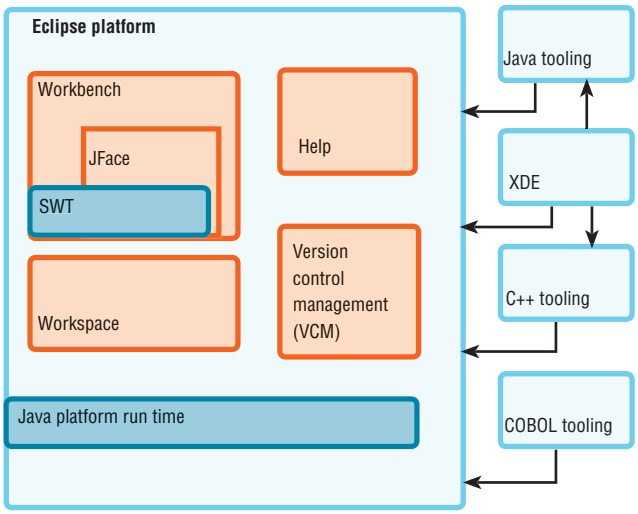

Figure 8. The Eclipse plug-in architecture.

In some respects, the Eclipse project and framework resemble efforts at creating common tool integration infrastructures of the past. However, Eclipse has a number of important differences that make it an excellent basis for the IBM Software Development Platform:

- High demand for a common Java desktop. *There are an estimated three million Java developers worldwide. They require a configurable Java development tooling to support their varied needs. The Eclipse project has been very successful in attracting attention and support in the Java community, with over 18 million downloads of Eclipse software. Hence, rather than creating technology in search of an audience, the Eclipse platform is widely welcomed by the Java developer community.*
- Flexible plug-in architecture. *Great attention has been directed in Eclipse at an extensible architecture based on plug-ins. This has proved very successful, with hundreds of projects building Eclipse plug-ins. Many of these projects are open-sourced; however, there ise also a growing number of commercial products based on Eclipse.*
- Use of open source. *The Eclipse technology is based on a broad range of existing open source technologies that are widely in use in the developer community, and hence, both very familiar and open to scrutiny by the developer community. Examples include Ant, JUnit, Xerces, and so on. This offers an immediate familiarity to developers.*
- Eclipse is an open standard. *As well as using open source technologies, Eclipse is also contributing to the open source community. The Eclipse software is made available under the Common Public License (CPL). The work on Eclipse is governed by a not-for-profit foundation consisting of members from many different organizations. This encourages wide adoption, use and experimentation with Eclipse by both commercial and academic communities.*
- Completeness of functionality for tool integration. *The Eclipse platform is not simply a configurable data repository for tools, as was typical of many previous tool integration efforts. The Eclipse platform includes sophisticated capabilities for other key aspects of integration, most notably metadata management and user interface (UI) design. For example, Eclipse includes a UI framework comprising the Standard Widget Toolkit (SWT), and built on top of that the JFace services providing standard ways to view and manipulate common UI elements. These are provided within a workbench that defines the overall structure of an Eclipse integrated software environment (IDE).*

- Broad industry support. *Many major software companies (both software suppliers and users) are supporting Eclipse with membership in the Eclipse Foundation, contributions of software and through use of Eclipse as a key part of their technology infrastructures. More than 175 vendors have, or plan to have, products based on Eclipse. This provides a maturity and breadth to the Eclipse technology that was never achieved in previous efforts.*

The role of Eclipse in the IBM Software Development Platform

IBM has played a significant role in Eclipse from its inception. IBM recognized the need for a powerful, flexible tool integration infrastructure for creating its next-generation software tooling platform. This focus on a single core technology platform for tools has been a central tenet of the IBM Software Group for the past five years, and significant progress has been made in delivering commercial software based on Eclipse, including the IBM WebSphere Studio family of products.

More recently, additional attention has been placed on the Eclipse infrastructure. With the recent additions of further tool offerings through the acquisition of IBM CrossWorlds®, Holosofx® and Rational software, IBM Software Group has extended its software tooling capabilities across a broader range of roles, projects and domains. Many of the tools that were acquired already included some level of interoperation with core IBM tooling (for example through common data formats, or through import and export using standard interchange mechanisms). However, it was essential that these tools coalesced around a single, clear technology platform that addresses the key customer needs for integration, flexibility and extensibility. The Eclipse technology plays that role in the IBM Software Development Platform, and has enabled significant acceleration of IBM's goal of a rich, highly integrated platform for software development. In this regard, it is impossible to overstate the importance of the Eclipse framework technology on the different teams contributing to the IBM Software Development Platform and the broader ecosystem that makes the IBM Software Development Platform viable to such a broad constituency of users.

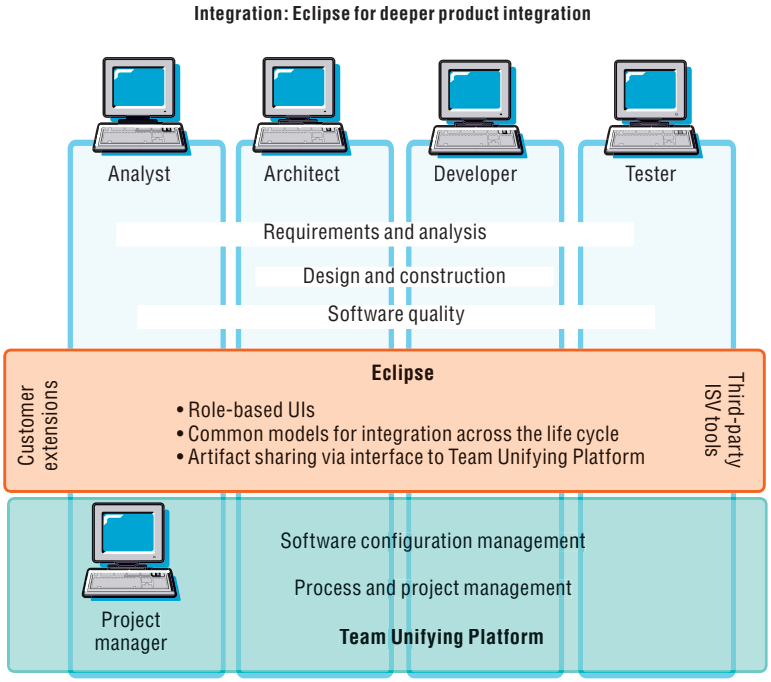

Figure 9. The key role of Eclipse in the IBM Software Development Platform.

As illustrated in Figure 9, the Eclipse platform performs three primary functions in the IBM Software Development Platform. First, Eclipse provides the UI framework and set of services that will be common across the IBM Software Development Platform. This offers a rich client experience and a large measure of visual consistency when moving between activities within the IBM Software Development Platform. Second, sharing of information across different activities is enhanced through the use of a set of common models expressed in the EMF technology. Third, a consistent set of team infrastructure capabilities is used within the IBM Software Development Platform. These are integrated with the Eclipse infrastructure as another plug-in, and hence, are available to all other plug-ins within the IBM Software Development Platform.

The Eclipse Modeling Framework

For integration among software tools to be meaningful, there needs to be common agreements on many of the underlying artifacts and processes that they share. These detailed shared semantics are represented using the Eclipse Modeling Framework (EMF). EMF is a modeling framework for Eclipse. EMF is a framework and code-generation facility that is typically used when defining the data structures manipulated by an application. EMF takes a model defined in UML, an XML (Extensible Markup Language) schema or Java interface, and generates the corresponding implementations classes. One of the key roles of EMF is to relate modeling concepts directly with their implementation. This brings to Eclipse the benefits of modeling with a low cost of entry for code-focused developers.

EMF is intended to unify the representation of the "data structures" defined in the application so that it is irrelevant whether these structures are defined in UML, as XML schema or as Java interfaces. For example, to create an application to manipulate an XML message structure, an XML schema can be defined. A UML class diagram for that schema can then be generated using EMF. Additionally, a set of Java implementation classes for manipulating XML can be generated. Similarly, starting with Java code describing the key interfaces in the design for an application, the corresponding UML model and XML message structure can be generated using EMF.

The models described in EMF are represented in an internal model called *Ecore*. EMF is IBM's realization of the Meta Object Facility (MOF). MOF is a standard defined by the Object Management Group (OMG) for describing metadata repositories. MOF defines a subset of UML for describing class modeling concepts within an object repository. Hence, MOF is similar to Ecore in its ability to specify classes with its structural and behavioral features, inheritance, packages and reflection. Where they differ, however, is that MOF has additional complexities for the life cycle, data structures, package relationships and complex kinds of associations.

The open MDD Platform built on Eclipse

Having read a description of the key elements of Eclipse, you can now see how important a role Eclipse plays in defining the next-generation IBM Software Development Platform. The Eclipse platform provides an open, extensible tools framework offering a rich client experience, a plug-in architecture for ease of extensibility and a sophisticated modeling framework for deep levels of semantics integration.

Then, to leverage this platform, the integration process involves the following steps:

- *Technology or tool-specific metamodels are defined using EMF. Where possible these are based on industry-standard metamodels (for example, UML) and extended where necessary.*
- *Much of the infrastructure for tool integration is generated from the EMF models. As the tools and their integration evolve, the infrastructure is regenerated from the models as necessary.*
- *Common metadata semantics are refined and shared (defined using UML, XSD, XMI, Annotated Java, and so on).*
- *A common programmatic interface is used for all tool interaction (Java).*
- *A common metadata interchange approach is used across the tools, and for external interaction with partner tools (XML).*
- *Simple implementation features are generated from the models (CRUD operations, basic editing capabilities, and so on).*
- *Metamodels for J2EE, Web services and model-driven architectures are used to drive internal transformations.*

The resulting technical infrastructure of the IBM Software Development Platform, as illustrated in Figure 10, consists of three primary components.

- *At the heart of this infrastructure is a set of open source technologies provided by the Eclipse project. This includes the Eclipse core, the various plug-ins and a set of metamodels defined in the EMF.*
- *IBM value-add capabilities are built on top of this set of open source technologies. These are leveraged across the IBM portfolio and they provide a range of reusable services to IBM engineering teams.*

- *Underpinning all of these capabilities is the team platform. This consists of the core IBM technologies for data sharing, artifact management, team interaction and information aggregation.*

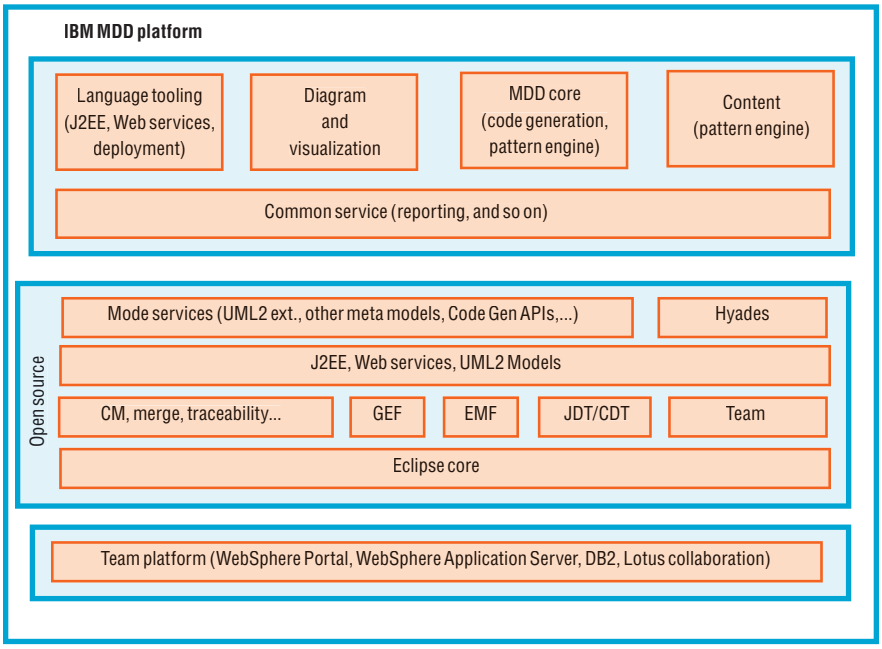

Figure 10. Details of the IBM Software Development Platform technical infrastructure.

Two technology developments in particular illustrate the power and openness of the IBM Software Development Platform based on this technical infrastructure: the UML2 metamodel and the Hyades testing framework.

UML2

The UML2 metamodel is a realization of major elements of the UML 2.0 specification in EMF. This will be the primary realization of UML 2.0 underlying all tools in the IBM Software Development Platform, and will hence represent the core semantic elements underlying the IBM Software Development Platform. It is used extensively in the various IBM tools, and extended where necessary by each tool through the standard UML extension mechanisms.

The UML2 metamodel is being developed by an open source project within the Eclipse community. The objectives of this project are to provide a useable implementation of the metamodel to support the development of modeling tools, a common XML Metadata Interchange (XMI) schema to facilitate interchange of semantic models, test cases as a means of validating the specification, and validation rules as a means of defining and enforcing levels of compliance.

The UML 2.0 specification is currently undergoing finalization. The goal is to create a first full release of the UML2 metamodel to coincide with release of the final specification, and the release of Eclipse 3 (Summer 2004).[4]

Hyades

The Hyades Testing Framework provides a single view of test assets and the testing life cycle for all tools in the IBM Software Development Platform. It facilitates integration of test activities throughout the life cycle, encouraging test-first approaches to development, and enhancing traceability from test artifacts to other artifacts in the life cycle. Hyades is designed to support a full range of testing methodologies via an open source infrastructure layer through which test and trace tools interoperate.

The Hyades Testing Framework is developed via an open source effort within the Eclipse community. The goal of the Hyades project is to integrate test and trace tools into the Eclipse environment to enable compatibility with tools across the software life cycle. This integration will reduce the cost and complexity of implementing effective automated software quality control processes. For developers, Hyades will improve solutions for both functional and performance-related testing by helping improve interoperability and lowering the cost of tools acquisition and subsequent cost of ownership.

Hyades uses UML to describe all artifacts used during testing (for example, traces and tests). To provide interoperability of tools and concepts, the test artifacts take a form that is defined by the OMG-supported testing profile. The artifacts are also compatible with MOF so that they can be stored and retrieved through EMF.[5]

Summary of technical underpinnings

IBM is creating a sophisticated set of services as the basis for the IBM Software Development Platform. Based on open standards, this set of services will help provide consistency and uniformity across IBM's tools, and provide openness for partners and customers to access those tools and extend them with value-added services.

IBM is also contributing metamodels and tooling frameworks into the open source. The UML2 metamodel and the Hyades Testing Framework are two examples illustrating the power of this approach. They both leverage open source technologies in their creation and use, and contribute back into the open source community to enhance tool integration across the IBM Software Development Platform. Both the UML2 and Hyades technologies are freely available to the software community in the Eclipse project.

WebSphere programming model

A key aspect of the IBM Software Development Platform is the use of a programming model influenced strongly by a service oriented architecture that is being implemented in the WebSphere platform, the IBM middleware stack (DB2, Tivoli and Lotus), and in particular, within the IBM Software Development Platform.

Key elements of the programming model common to IBM Software Development Platform and the IBM middleware platform include:

- *Service Data Objects (SDOs), now in the standardization path at the Java Community Process. SDOs provide a simplified data access programming model for various resources (data as well as EIS) and complement the core Web services standards XML, Web Services Definition Language (WSDL), and Simple Object Access Protocol (SOAP).*
- *BPEL4WS, which is a service orchestration and component scripting standard that supports workflow and business process integration.*

- *JSF, which is a Java framework that speeds Web application development for developers who are not expert J2EE developers.*
- *Customization of applications using external policies and rules. A series of emerging standards are in development for policy definition and enforcement, including Web Services Policy and OMG Business Semantics of Business Rules (BSBR).*

A role-based solution portfolio

IBM leverages the IBM Software Development Platform to create a portfolio of solutions targeted at key roles within the software development life cycle. For example, in the area of design and construction, there are a number of specific offerings available to practitioners. These solutions are illustrated in Figure 11.

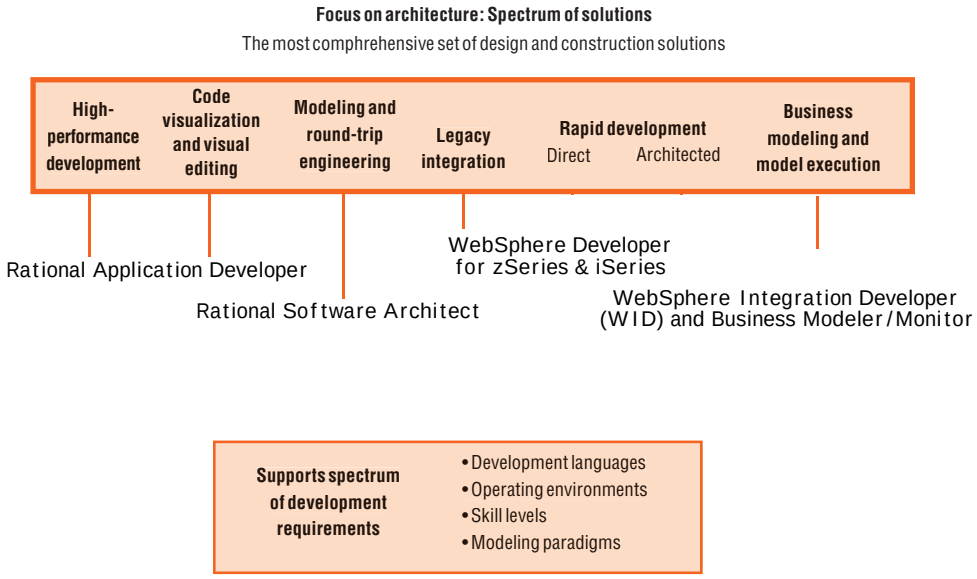

Figure 11. An Example of the role-based solution portfolio.

In Figure 11, we see that design and construction offerings address a range of practitioners concerns. At one extreme, we have code-focused practitioners expecting highly productive, code-oriented tools that help them drive the IBM run-time platforms. At the other extreme, we have business-focused practitioners needing to express business concerns in the language familiar to business-oriented communities.

In all of these areas we see an aggressive migration to the Eclipse-base technology infrastructure described before. The first wave of tools leveraging this infrastructure is already in commercial use, most notably the WebSphere Studio products and the Rational Rose XDE products, illustrated in Figure 12.

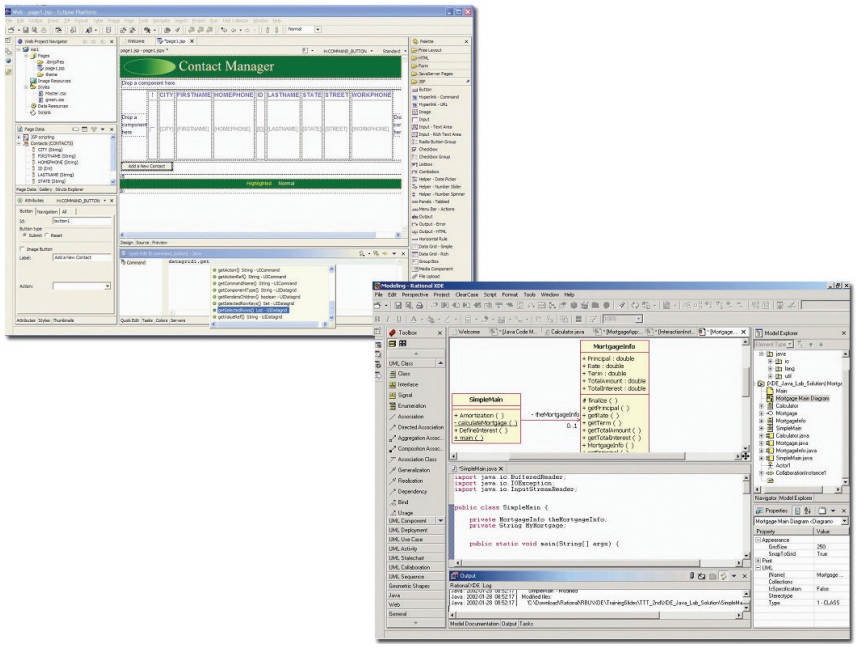

Figure 12. Examples of released products leveraging the Eclipse platform.

Additional tools built on this platform are already in alpha and beta stages, and are planned for release through 2004 and 2005.[6] These tools capabilities will expand the openness, completeness and integration of the IBM Software Development Platform in the coming months

Summary

The role of software in the many businesses is now seen as central to their ability to compete effectively and efficiently. Focusing attention on software as a core business in those organizations will help to channel investments into IT toward innovation in key business processes that differentiate the organization from its competitors. The IBM Software Development Platform plays an important role in helping organizations to create a set of services capable of realizing this goal.

In this paper, we have described some of the ways that the IBM Software Development Platform contributes to realizing that value. In particular, we have illustrated how that platform can be realized today with current technologies. Furthermore, we have highlighted the technical infrastructure that forms the core of the IBM Software Development Platform. IBM is aggressively investing in the future of these technologies, leveraging those investments in the solutions delivered to customers.

Further reading

Gallardo, D. et al., *Eclipse in Action: A Guide for the Java Developer*, Manning Publications Company, 2003.

Budinsky, F. et al., *The Eclipse Modeling Framework*, Addison Wesley, 2003.

Booch, Grady, James Rumbaugh, and Ivar Jacobson, *The Unified Modeling Language User Guide*, Addison Wesley, 1998.

Brown, A.W. et al., *Principles of CASE Tool Integration*, Oxford University Press, 1994.

Messerschmitt, D.G. and C. Szyperski, *Software Ecosystem: Understanding an Indispensable Technology and Industry*, MIT Press, 2003.

IBM Corporation
Software Group, Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
04-04

[1] See the World Information and Technology Services Alliance (WITSA) report, “Digital Planet 2002: The Global Information Economy”, available at www.witsa.org/dp2002execsumm.pdf

[2] See Bob Evans, "Business Technology: IT is a must, no matter how you view it", ***InformationWeek*** May 19, 2003, www.informationweek.com/story/showArticle.jhtm?articleID=10000185

[3] This high-level example is drawn from a number of engagements by IBM in the Travel and Transportation industry sector. Contact Ben Amaba (baamaba@us.ibm.com) for further details.

[4] For further details, see www.eclipse.org/uml2

[5] For further details, see www.eclipse.org/hyades

[6] All such features are subject to change, and are used here purely for illustrative purposes.