A New Development Environment for the Enterprise: Rational Software and the Eclipse Platform

Sky Matthews

TP560, 12/02

Rational®
the software development company

Overview

Integrated Development Environments (IDEs) are the tools that developers work with every day to construct and maintain software. IDEs have evolved dramatically in the past few years, and some recent innovations have changed the way we think about our development tools. The IDE business is highly competitive and constantly changing. It is hard to look past the usual marketing rhetoric to see the broader trends and fundamental drivers that may affect the IDEs and their success in the market. In this paper, we provide an overview of emerging trends that are driving the evolution of IDEs, and draw some conclusions about what the IDE of the future will look like.

We define the next logical evolutionary step in the IDE business as being the formation of what we call the “Enterprise Development Environment”. An Enterprise Development Environment is one that goes beyond providing tooling for individual technologies and development activities and enables the efficient construction of enterprise-level software. An Enterprise Development Environment provides integrated tools for a wide variety of technologies, and supports all the major activities of the software development lifecycle.

The past 12 months have seen the introduction of two tool environments that have the potential to be Enterprise Development Environments, IBM’s Eclipse platform and Microsoft’s Visual Studio .NET. IBM has donated the Eclipse platform into open source, and the growth of that platform is now managed by Eclipse.org, a consortium of independent software vendors (ISVs). Rational is a founding member of Eclipse.org and is contributing to the growth of the Eclipse platform and tightly integrating a wide variety of class-leading tools into Eclipse. Through this paper, we identify the requirements of an Enterprise Development Environment, and outline the steps that Rational is taking to help turn Eclipse into an Enterprise Development Environment.

What is Eclipse?

Eclipse is a tool platform – a framework for building and integrating development tools. Eclipse is intended to be language-independent. It can be used as a development environment for tools for many different languages, as well as for tools that are language-independent.

The Eclipse platform includes a number of frameworks and components required for constructing typical development tools. See http://www.eclipse.org/eclipse/index.html for more information. It is essentially the shell of an integrated development environment, together with various building blocks for adding functionality inside this shell (menus, editors, tool windows, toolbars, drawing surfaces, help, wizards, etc.).

Eclipse is being used to build tools for Java, J2EE, web services, XML, HTML, C/C++, Cobol, Aspect programming, UML, test tools, and more. In addition, a variety of team productivity applications, such as version control and change management, are integrated with Eclipse. Eclipse runs on a wide variety of operating systems, including Windows 98/ME,/2000/XP, Solaris, HP-UX, AIX, QNX, and Linux.

What is eclipse.org?

Eclipse.org is the organization that builds and maintains the Eclipse platform and various surrounding projects. Eclipse.org was founded by a broad consortium of companies for the purpose of enhancing the Eclipse platform and promoting it in the software community. Eclipse.org is open to membership by any organization that agrees to abide by the requirements, namely releasing products based on Eclipse, and contributing to the enhancement of the Eclipse platform and its promotion. Membership applications are subject to approval of the Eclipse.org board.

The Evolution of the Integrated Development Environment

To understand why Rational has embraced the Eclipse platform, it is important to see how Eclipse fits into the larger trends around software development tools.

The first integrated development environments appeared in the early ‘90s, consisting essentially of an editor, compiler and debugger. Subsequently, development environments grew in size and sophistication. Customers began demanding that other vendors integrate their tools with the IDEs, giving the customer the ability to use a best-of-breed set of tools.

IDEs are now closely coupled to the run-time platforms for deployed applications (virtual machines, web servers, application servers, databases, etc.). In fact, the very business model of the IDE has changed – the software platform vendors (IBM, Microsoft, Oracle, etc.) use tooling as a way to differentiate and sell the platform. Part of the strategy of a platform vendor is to create a partner ecosystem around their development tools and run-time platform. To enable this, most of the leading IDEs now have extensive APIs for 3rd party tool integrations. To support such an ecosystem, the IDE vendor must be of sufficient size and market share and must be motivated to ensure success of many other 3rd party software vendors.

In this environment, only two kinds of IDE platform will survive in the long run: those that are supported by the largest software platform vendors (namely Microsoft and IBM), and those that are supported by the open source community. The industry is now undergoing a consolidation, and Rational believes that two primary IDEs will dominate: Microsoft's VisualStudio .NET and Eclipse. VisualStudio has long been the benchmark IDE for Windows development, and VisualStudio .NET includes extensive APIs for integrating 3rd party tools. For non-Microsoft technologies and environments (Java, UNIX/Linux, embedded systems), Eclipse has the advantage of being both the core of IBM's IDE family (WebSphere Studio) and supported by the open source community. As these IDEs achieve a critical mass of users and ecosystem partners, they will be able to provide compelling value spanning multiple roles, multiple languages, and multiple technologies in use across an entire enterprise. These IDEs have expanded beyond the narrow scope and market limitations of other IDEs to become true Enterprise Development Environments. Figure 1 illustrates the evolution of IDEs over time, culminating in the realization of an Enterprise Development Environment.

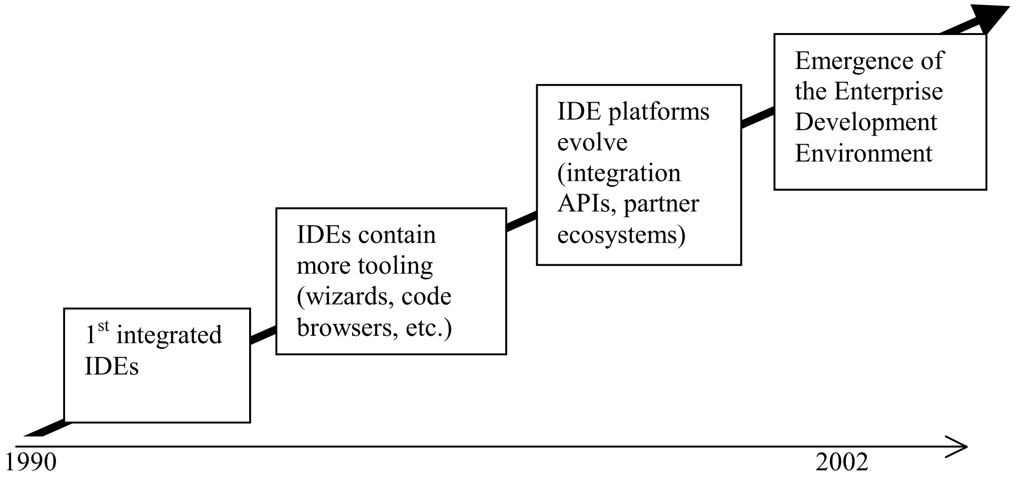
1st integrated IDEs
IDEs contain more tooling (wizards, code browsers, etc.)
IDE platforms evolve (integration APIs, partner ecosystems)
Emergence of the Enterprise Development Environment
1990
2002

Figure1: IDE evolution over time

An Enterprise Development Environment

The successful IDE of the future will be one that can evolve to become an Enterprise Development Environment; that is, a tool environment that:

- is open and allows vendors and customers to integrate different tools and enable them to interoperate
- can support many different roles and tasks within the development team, from analysis and design through coding, debugging, deployment and testing
- can support a heterogeneous environment, wherein users need to work with a mixture of programming languages, operating systems, run-time platforms and development tools

Support for tool integrations

Historically, the world of software development has been highly fragmented, with a large number of tools in use, each having its own proprietary API. Motivation for vendors to integrate their tools with those of other vendors has been low and no integration standards exist. The result is that vendors and customers alike are frustrated. Customers are frustrated because integrations are incomplete or non-existent between any two tools, requiring the customers to use workarounds or invest their own time and money in developing and maintaining the integrations. Vendors are frustrated because developing point-to-point integrations with other tools is costly and difficult to maintain, and detracts from new feature development. Figure 2 shows all of the potential integration points between different tools in a typical environment. It is rare that independent tools from different vendors would achieve such a level of integration, without a common underlying set of APIs. An Enterprise Development Environment is built on a platform that contains a complete set of APIs spanning all these different functions. Further, these APIs must be part of either an open standard or a de-facto standard in order to have sufficient support to make them viable.

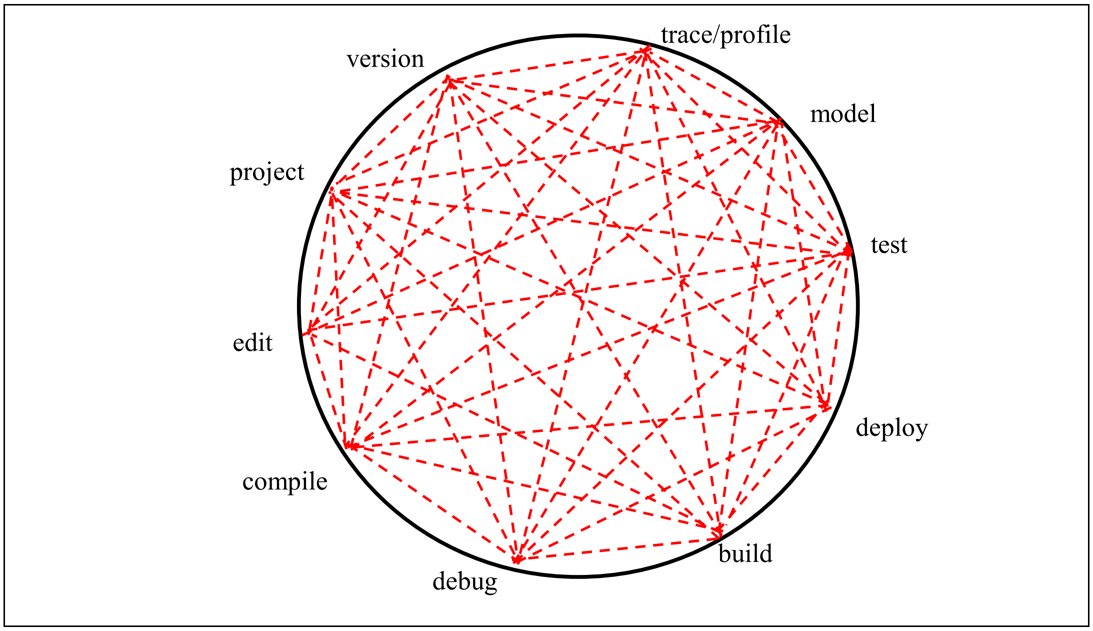

Figure 2: Components of a development toolchain and their potential integration points

Support for teams

Software development is a team activity, and it works best when all members of a team are readily able to share assets, use a consistent process, and a common set of tools. Many teams today have a complex environment consisting of many different tools, often lacking integrations, and making it difficult for different roles to communicate and interact. Rational has long been at the forefront of unifying software teams by integrating the tools for different activities in the software lifecycle. Since most teams revolve around the software development activity, the next logical step is to bring the team unifying tools closer to the development activities. The vision for the Enterprise Development Platform enables us to do that. It should be possible to bring together support for all the different roles in the software development process (analysis, design, development, testing, and so on) on a common tool platform if the platform is extensible enough. This will allow individuals, who often play multiple roles, to switch seamlessly between activities without having to switch between different tool environments. Furthermore, such a platform will improve the ability for different team members to share and access information.

Support for run-time platforms

Off-the-shelf run-time platforms and middleware are increasingly becoming a key part of many applications. This trend will continue to grow with the adoption of web services. While no environment may be able to support all of the run-time platforms in use today, it is important that the development environment contain deep support for the key platforms in use in a particular enterprise. The environment must be open and flexible enough to allow other vendors and even customers to build integrations for other run-time platforms.

The same is true of operating systems. While Windows does dominate the desktop environment, many application deployment environments include UNIX servers, and many software development teams do use UNIX systems on the desktop. There has also been significant growth of Linux on both the desktop and the server side. An ideal development environment would support development and deployment on a wide variety of hardware and operating systems.

Benefits of Eclipse

Eclipse is the first development environment that truly offers the potential of being an enterprise development environment for non-Microsoft technologies – that is, a single IDE supporting all the various software development needs for the whole enterprise.

There are several key reasons for this:

1. Eclipse is a flexible framework that, through its extensibility, can support many languages and many target run-time platforms
2. Eclipse is open source, and freely downloadable
3. Eclipse runs on a variety of desktop platforms (Windows, UNIX, Linux, Mac)
4. Eclipse is supported by a large community of software vendors
5. Eclipse supports all the roles across the software development lifecycle
6. Eclipse has an extensive and highly robust set of APIs
7. Eclipse is the basis of the IBM WebSphere Studio family of development tools, and through those tools provides extensive support for the leading Java run-time platform

There are many benefits provided by adopting Eclipse:

1. Support for many roles and tasks in the software development lifecycle
2. Easier transition of practitioners to new projects
3. Much higher degree of tool interoperability
4. Lower cost of environment setup and maintenance for customers
5. Lower cost of integration for vendors

Eclipse offers a path away from the complexities of tool integration because its APIs are an open standard, defined and supported by a consortium of vendors spanning many different tool categories and business domains. Vendors need only integrate with a single set of APIs rather than different proprietary APIs from other vendors. Customers can feel more confident that tools from different vendors will interoperate as expected within Eclipse. Figure 3 identifies the core platform and some of the extension APIs that will enable Eclipse to become an Enterprise Development Environment.

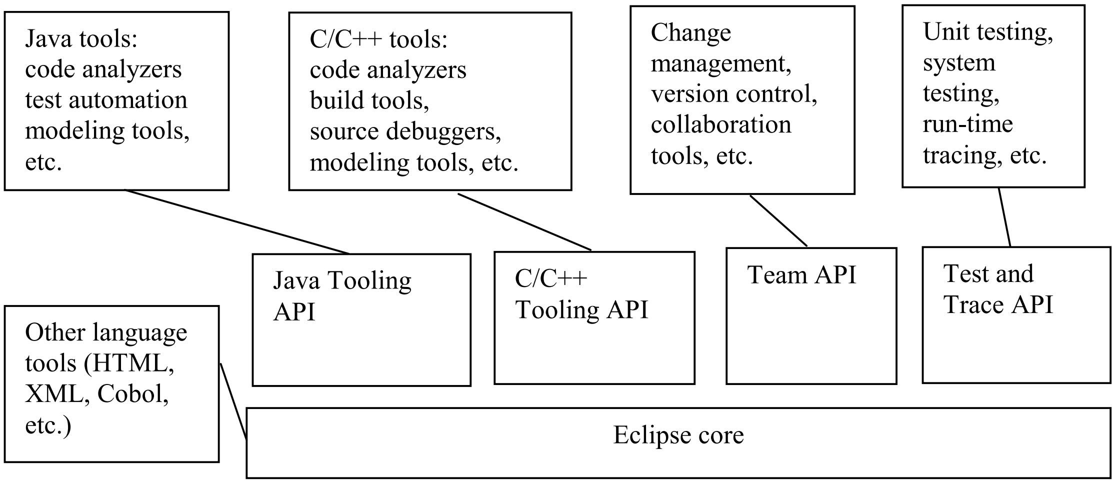

Figure 3: Components of a development toolchain and their potential integration points

Rational's Eclipse-based Product Line

The Eclipse IDE platform is a key part of Rational's product strategy. Rational is a founding member of the eclipse.org community. Rational has demonstrated its commitment to Eclipse by releasing several products that build on or integrate with the Eclipse platform since February 2002:

- Rational XDE, the visual modeling tool for Java
- Rational RobotJ, the java-based functional testing tool shipping in Rational Suite TestStudio
- Rational ClearCase, the industry-leading change management tool

Rational will continue and expand its leadership in the Eclipse community through contributions to various projects on Eclipse.org, support for Eclipse.org foundation activities, and, of course, additional Eclipse-based product releases in the coming months.

Rational XDE Java Professional Edition: the Rational eXtended Development Experience (XDE) for Java provides a frictionless design and development environment built around the Eclipse platform. Rational XDE provides visual design and development tools that are tightly integrated with IBM® WebSphere giving the developer a single user experience. Rational XDE Professional supports developers in building applications, including web-centric solutions. The blend of features (e.g., roundtrip engineering of code, automatic or on demand synchronization of code and models, pattern engine support, multi-model support, reverse engineering of database assets into data models, community, and selective process guidance) reduces the effort required to comprehend and develop code and designs.

Rational ClearCase: the Rational ClearCase Plugin provides full access to Rational's market- leading configuration management tools from within Eclipse. The ClearCase plugin supports the complete family of ClearCase products, including support for advanced ClearCase capabilities such as Unified Change Management, and runs on Windows and Linux. The ClearCase plugin makes ClearCase operations (checkout, checkin, deliver, rebase, etc.) accessible directly from inside Eclipse. The deep integration of the ClearCase plugin with the core

Eclipse Team UI enables several unique new capabilities within Eclipse, such as the ability for the ClearCase plugin to detect edits on files under ClearCase version control and automatically check them out.

RobotJ in Rational Suite TestStudio: RobotJ is a functional testing tool for automated testing of java and Web applications that ships in Rational Suite TestStudio and Rational TeamTest products. RobotJ uses industry-first technology for automating functional testing that absolutely minimizes script maintenance by generating resilient, reusable scripts and intelligently differentiating between a bug and a simple data change. Testers are freed from tedious maintenance rework, and can begin effective interface testing before the design is finalized.

Rational's Contributions to Eclipse

Rational is a founding member of Eclipse.org, and actively participates on the Eclipse.org board of stewards and its working subcommittees. Rational is also making significant technical contributions to the Eclipse core and to the C/C++ Development Tooling (CDT) and Hyades automated software quality projects.

Core platform

Rational provided the port of Eclipse to Solaris and HP-UX. In addition, Rational has contributed to the development of the SWT graphic library for the UNIX platforms. Rational has worked closely with Eclipse.org to perform testing, log bugs, provide recommended bug fixes, and refine the definition of various Eclipse extension points since it's initial release.

C/C++ Development Tooling (CDT)

The CDT (C/C++ Development Tools) project is working towards providing a fully functional C and C++ Integrated Development Environment (IDE) built on the Eclipse platform. It adds a C/C++ Perspective to the Eclipse Workbench that supports C/C++ development with a number of views, wizards, a powerful editor, and a debugger. The CDT is designed to provide an extensible architecture that will provide support for integration of tools provided by independent software vendors. Default implementations of all interfaces and extension points will be supplied for various platforms.

Rational is contributing to the development of interfaces to simplify the integration of C/C++ tools into the IDE. The CDT will be configurable to support many different build environments and tools. Compiler vendors, 3rd party tool companies, and end users will be able to extend the CDT to support other C/C++ tools. Rational is also contributing expertise on code parsing to provide capabilities such as class browsers, searching and refactoring, making Eclipse into a top-notch IDE for C/C++ development.

Hyades – Automated Software Quality for Eclipse

The Hyades project framework will make it possible to integrate a broad range of functional verification, quality assessment and load testing tools with the Eclipse Platform's workbench and other tools. Project Hyades includes several extensions to the core Eclipse Platform which ease the integration of tools needed for answering deployment issues commonly encountered with today's end-to-end development projects. Hyades is the first of many projects that will lead to establishing an "I*E", or integrated "Everything" environment that take development support beyond the confines of a traditional IDE.

Hyades will implement an OMG defined UML testing profile in which Test Case, Test Trace, Test Objective and Verification artifacts are maintained. Test objectives related to performance and scalability can be evaluated for a range of real world deployment environments, including alternate mixes of server and network interconnection technologies. New approaches to quality assessment testing that link the initial requirements definition, object oriented model of the application, and test management evaluation tools will also be supported.

Rational will be an active contributor of leadership, design, and source code implementation to the Hyades project.

Summary

The next logical step in the evolution of the Integrated Development Environment concept is the Enterprise Development Environment, a tool environment that can support the complete software development lifecycle, can support many different languages and run-time technologies, and can attract a strong community of tool vendors. There are really only two IDEs that meet these requirements: Microsoft VisualStudio .NET, because of the dominance of the Microsoft platform, and Eclipse, because of its fundamentally open nature and its relationship to IBM, the leading Java platform vendor. In this paper we have explored what makes Eclipse unique in its support for enterprise-level development, and how adopting Eclipse will benefit customers. Finally, we have highlighted Rational’s commitment to the Eclipse platform and enumerated the key contributions that Rational is making to the Eclipse project.

Rational Software
Dual Headquarters:
Rational Software
18880 Homestead Road
Cupertino, CA 95014
Tel: (408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
Tel: (781) 676-2400

Toll-free: (800) 728-1212
E-mail: info@rational.com
Web: www.rational.com
International Locations: www.rational.com/worldwide

[ELS1]

[rmr4]
[sky5]
[rmr6]
[rmr7]