



October 7, 2004

Eclipse Changes The Game For Development Tools

by **Carl Zetie**

with Liz Barnett and Carey E. Schwaber

EXECUTIVE SUMMARY

Eclipse isn't just changing the development tools that enterprises buy: It's changing the way that enterprises buy development tools. Eclipse's early appeal for IT organizations was the ability to acquire features for free or at low cost. Eclipse's real value comes from the ability to assemble just the right combination of tools and resources that an organization — and individual roles within an organization — needs. Eclipse is becoming more than a tools integration platform; it's becoming a way for IT organizations to reclaim control of their development environment. For development tools vendors, this shift in perspective presents a huge challenge to the traditional monolithic IDE and other comprehensive tools — but for those vendors brave enough to grasp the implications and embrace the change, it also represents a huge opportunity to change the economics of the development tools market.

RESEARCH CATALYST

Forrester interviewed and conducted numerous inquiries with IT organizations and leading development tools vendors.

TRADITIONAL DEVELOPMENT TOOL MODELS ARE BREAKING DOWN

IT's traditional model for development tools is based on an age-old tension between two mutually exclusive models:

- **Best-of-breed.** The organization assembles tools from multiple vendors and painstakingly integrates them.
- **Tightly integrated suite.** The organization buys into a single vendor's (and its close partners') vision, trading off the potential benefits of better tools in some areas against the promise of better integration.

However, this dichotomy hides another problem: Even supposed best-of-breed tools are rarely best in every respect. It is unlikely that any given IDE will have the best features in every area or even for the individual tastes of every user. Do you like IntelliJ's refactoring capability best? Then you have to forego IBM Application Developer's deployment features.¹ Want the debugging capability of Application Developer? Then you can't have JBuilder's visual layout tools.

For both users and vendors, adopting updated versions of IDEs has become a game of diminishing returns: Each new feature adds value for fewer users, adds more clutter to those users' desktops, and requires the same amount of development cost and vendor support. Out of this frustration is emerging the idea that the traditional model of the IDE (and other tools) is simply the wrong level of granularity at which to package products. The Holy Grail of development tools has always been tightly integrated best-of-breed tools — a quest that the Eclipse framework finally may be bringing into reach.

ECLIPSE GIVES DEVELOPERS A NEW WAY OF LOOKING AT TOOLS

When we interviewed more than 20 IT organizations, including Fortune 500 companies, midsize companies, and public sector organizations, about their development tools and practices, one tool dominated our conversations: Eclipse. What these organizations like about Eclipse is not just the potential low cost, but the fact that it puts them back in control.

- **Functionality comes in more granular components.** Instead of having to make all-or-nothing choices between entire IDEs, organizations can add specific units (plug-ins) that correspond to a particular requirement. The best plug-ins will do one job and do it well, whether that's persistence modeling, unit testing, refactoring, or enhancement of the code-editing experience. For many IT shops, individual plug-ins are a much more appropriate level of granularity for creating the ideal development environment.²
- **The IT organization decides what functionality it wants.** Eclipse offers unprecedented flexibility in choosing IDE functions. The GUI layout tool is an optional component — and you don't even have to use the basic Eclipse code editor if you prefer an alternative, such as a commercial product or even a simple text editor. Organizations can add whichever modeling tools, persistence framework, build tools or other plug-ins they need — and exclude those that they don't.³
- **Upgrades and enhancements come independently.** Each plug-in evolves at its own pace, not at the speed of a monolithic IDE's release cycle. Provided everything is compatible with the same revision of the framework, you're never left waiting for an enhancement in one area while an unrelated enhancement is completed.
- **Tool integration is driven by user demand, not by vendor alignments.** With open interfaces and common metadata, plug-in authors are free to create the integrations that make sense for them and their customers. Integration is not controlled by proprietary APIs or partnership agreements.

This shifting perspective is already starting to have an affect on how users value (or don't) traditional IDEs. Users are questioning the value of a mainstream commercial product when so much can be had for free or low cost, without the exclusivity of traditional IDEs. Even development managers in organizations that use a mixture of IBM Application Developer and Eclipse's IDE have told Forrester that the primary feature they miss about Application Developer is deployment to WebSphere. Although Application Developer includes a long list of capabilities not included in the base Eclipse IDE, for these users those features simply weren't of enough value. Everything they really needed was provided by the combination of the Eclipse IDE and various plug-ins. And we heard similar stories about all the leading IDEs. (Ironically, Application Developer is itself an Eclipse-based tool, and IBM has demonstrated more commitment to Eclipse than any other vendor.)

One other surprise emerged from our conversations: Administering an Eclipse-based tools environment composed from multiple independent plug-ins is far less of a burden than we expected or some organizations feared. While it's valuable to have a consistent Eclipse image across the development team, and it's worthwhile to have one person responsible for evaluating plug-ins and managing the image, it's far from a full-time job. A development manager at a large agricultural company told us that "even at peak periods, such as evaluating the upgrade to Eclipse 3.0, it's as little as one person-day a month, and considerably less in between such upgrades."

VENDORS MUST EMBRACE ECLIPSE

Tool vendors must respond to this new challenge: While there are still a lot of organizations that value the tight integration that a single vendor product or suite can offer, for many others flexibility and value are trumping integration and ease of configuration.

The flip side of this challenge is an opportunity: As Eclipse has commoditized the core elements of a basic IDE, vendors need to ask themselves what the real value that their products add is. The brave next step is then to be willing to offer just that valuable functionality as Eclipse plug-ins. For example, the organization mentioned above might be happy to buy IBM Application Developer's WebSphere deployment capability as a plug-in for an appropriate price if it could tightly integrate it with its preferred editors and modeling and testing tools. IBM Application Developer has many other features that would be of interest to some users, of no interest to others, and that would make good candidates for commercial Eclipse plug-ins.⁴

For tools vendors in the future, two approaches have a chance at success:

- **Comprehensive suites.** For user companies that want a complete, integrated environment, a life cycle suite still may be the best way to go. However, vendors need

to realize that the value of a suite lies more in the cross-tool process integration that it enables than in the excellence of individual tools.⁵ And suite vendors will need to recognize that even the best suite will not meet all the needs of all the users. Successful suites will be those that allow the buyer to add specialized plug-ins as well as replace suite components at will with preferred plug-ins, while still retaining the cross-tool automation, traceability, and reporting. Few vendors have a shot at making this model work: possibly Borland, IBM, and Microsoft.

- **Granular tools.** For users that don't embrace a single vendor's suite as the heart of their development life cycle, the best approach will increasingly be to assemble the functionality they need around an Eclipse core. And even those that do embrace a suite will want to add fine-grained additional plug-ins. Vendors can tap into this by breaking down monolithic IDEs and factoring out the parts that are most valuable, whether that's the richness of their refactoring or the depth of their debugging. For some tools, the right level of granularity might be as large as it is today (for example, version control); for others, small pieces of functionality will be more appropriate.⁶

Tools that fit neither of these models — tools that lie between these two approaches, that are neither comprehensive nor flexible, and that don't offer functionality at a granularity that matches what users value — will be increasingly pressured by the these two approaches. It won't happen overnight, but the rapid growth in popularity of Eclipse and the growing scope of plug-ins should sound alarm bells for vendors.

There are already a few good precursors of such a model in place. In the realm of object persistence and caching, smaller vendors like Persistence and Thought, Inc. have produced Eclipse plug-ins for their object/relational products that can complement modeling and development tools.⁷ GUIs supporting frameworks like Hibernate and Spring are offered separately from the frameworks themselves (Hibernate in particular seems to have generated a subecology of supporting tools of its own). And in J2ME development, companies like IBM and Nokia offer plug-ins that add J2ME support to an existing Eclipse IDE, rather than reinventing the IDE just to add J2ME support. In fact, IBM's approach to J2ME offers a glimpse of the future: IBM offers separate plug-ins for basic J2ME development (WSDD) and sophisticated enterprise-integrated J2ME applications (WME).⁸ Another example of the model comes from outside the Eclipse world: JetBrains offers a complete IDE with superior refactoring capabilities for the Java world (IntelliJ) and makes this refactoring functionality available to the C# world as a Visual Studio plug-in (ReSharper).

WHAT IT MEANS

WITH GREAT POWER COMES GREAT RESPONSIBILITY

IT organizations will gain a new degree of control of their development environments in terms of both the content and the pace of evolution. But they must be prepared to take responsibility accordingly: Assign someone the responsibility for evaluating, acquiring, and integrating new plug-ins and distributing them to developers. Stay current with available plug-ins, and monitor the viability of those that you most depend on. Plan major upgrades — particularly those involving a revision of the underlying Eclipse framework — with great care. Also, decide how much conformity is necessary: Does everybody on a project team, in a division, or across the entire company need the same collation of Eclipse plug-ins, and what is the cost of tolerating more than one image? Early experiences suggest that, at a minimum, the project team is the right level to standardize.

For vendors accustomed to selling enterprise IDEs, this shift represents a dramatic alteration in the way they develop, price, and market tools, as well as in their relationships with their customers. On the one hand, they will be freed from the tedium of recreating core IDE elements, as well as the treadmill of feature-level comparisons. On the other hand, they will have to get used to a sales model that forgoes owning the customer relationship and adopt pricing and distribution models that reflect the value of each available plug-in rather than the cost of an entire enterprise IDE.

ENDNOTES

- ¹ IBM Application Developer was previously known as WebSphere Application Developer (WSAD).
- ² The adoption of more granular development tools environments mirrors a trend to adopt lighter-weight deployment frameworks. In many cases, the same organizations that are adopting Eclipse are also composing their own deployment environments from frameworks like Hibernate, Spring, and Struts. Like Eclipse plug-ins, the frameworks each do one job well and bring parallel benefits to those that Eclipse offers.
- ³ Eclipse's three levels of integration — user interface, tool-to-tool automation, and shared metamodels — are what finally make this assembly approach to development tools a realistic possibility. For more information on the Eclipse model, see www.eclipse.org.
- ⁴ IBM Application Developer features include a JSF editor, SDO support, a visual editor for Struts, portal support, J2EE wizards, and Enterprise Generation Language for non-Java Web development, as well as tools for test, performance profiling, and code analysis. Many of these features are well suited in scope of functionality and breadth of appeal to be independently available as plug-ins for those users that value them.

- ⁵ The next generation of Visual Studio, currently in beta release, gives a glimpse of what such an environment might look like, as well as its potential benefits. Microsoft Visual Studio is not an Eclipse-based environment; however, the innovations in the next generation of Visual Studio and the forthcoming Team System reflect many of the same trends and user requirements. In the Microsoft world, however, there is no equivalent to the all-Eclipse model: Users adopting Team System will invariably get the bulk of their tools from Microsoft and fill in the gaps with third-party tools. See the June 14, 2004, Trends “Microsoft Visual Studio 2005: More Than The Sum Of The Parts.”
- ⁶ This granular model could also affect the data modeling tools market. Some vendors, such as Sybase, have already factored out the DDL generation so that updates to support new database releases can be provided out-of-band from product functionality releases. Other vendors still leave their users waiting for such updates while the rest of a conventional product release is completed. Often, buyers are forced to compromise in their tools selection between product features and generation targets. The logical next step is for data modeling tools vendors to allow interested third parties to develop DDL generators that use Eclipse metadata integration to completely decouple product features from target database versions supported. In this way, even the most obscure databases could be supported by the most mainstream modeling tools.
- ⁷ Persistence is currently being acquired by Progress Software Corporation and will become part of its ObjectStore operating unit. See http://www.persistence.com/news/press_releases/2004.09.27A.html for details.
- ⁸ WSDD is the “device” edition of IBM’s WebSphere application development tools. See <http://www-306.ibm.com/software/wireless/wsdd/> for details. WME adds integration with the Everyplace edition of IBM’s MQ and DB2 products. See <http://www-306.ibm.com/software/wireless/wme/> for details.