

# Jazz and the Eclipse Way of Collaboration

**Randall Frost**

**T**wenty-five years ago, computer programmers often wrote code in environments reminiscent of the cells of cloistered monks. But now that software development has become highly collaborative, developers tend to spend more time interacting with their colleagues than stringing together lines of code.

In the past, software collaborative efforts were more ad hoc than formal. So, collaboration strategies sometimes created more problems than they solved. In a typical workday, the need to respond to emails, attend meetings, participate in group discussions, and manage source control could take up so much time that little time was left to do any coding.

To improve collaboration in software development teams, IBM Research and IBM Rational software engineers have been working on the Jazz project. Jazz sets out to define a vision for the way products can integrate to support collaborative work, and to create a technology platform on which to build products to deliver on this vision. It focuses on developing better team-building strategies, managerial processes, architectural designs, collaborative coding techniques, and software development practices.

Jazz aims to be a scalable platform that can integrate tasks across the software life cycle. As such, it will consist of a set of plug-ins to Eclipse, IBM's Java-based, extensible open source development platform. Unlike Eclipse, however, which seeks to increase individual programmers'

productivity, Jazz aims to make teams and teams of teams more productive. By building on prior experience with Eclipse, Jazz's developers hope to address IBM customers' evolving application-development requirements.

## **A brief history**

The ideas behind Jazz have been around for quite a while in research milieus. It's only recently that IBM and others have paid serious attention, probably owing to the proliferation of social networking.

The Jazz project, which grew out of IBM's experience with Eclipse and other open source projects, has reportedly been under quiet development since even before IBM acquired Rational in 2002. An early visionary in the Jazz project was Li-Te Cheng of the IBM Watson Research Center, who coauthored a paper describing the idea in 2003 ("Building Collaboration into IDEs," *ACM Queue*, vol. 1, no. 9). In a preliminary version of Jazz, IBM Research attempted to integrate instant messaging into an IDE. But the present version has a much broader scope.

IBM showcased the present version at its Rational Software Development Conference last June in Orlando. According to IBM Rational's Global Communications office, this version is a response to increased interest in social networking and Web-based communities, as well as to customers' requests for help in managing software development as a strategic business process. Jazz's current incarnation is intended to

deliver better visibility, process guidance, artifact traceability, and management throughout the software development cycle.

Interestingly, the Jazz development team is using Jazz to build Jazz. According to the Global Communications office, the company has been self-hosting with Jazz since November 2006. Interactions with the Jazz community at jazz.net reportedly help the development team set priorities.

### Breaking new ground

According to Janice Singer of the National Research Council of Canada, software development collaboration has evolved via tools that control artifacts—pieces of source code, documentation, and reported bugs. “Jazz is trying to take that one step above,” she says.

Of course, software developers have been using collaboration tools such as email and instant messaging for a long time. Where Jazz breaks new ground is in incorporating collaborative tools into the integrated development environment (IDE)—right beside the editor, compiler, and debugger. (Microsoft’s new Visual Studio Team System is a similar effort.) This provides programmers a more seamless link between code development and collaboration.

As a provider of collaborative capabilities to small teams of developers, Jazz

- handles connections to the server infrastructure to support messaging and source control,
- places hooks in Eclipse to track developers’ interactions with source code and source control, and
- integrates the user interfaces that developers use to communicate with each other.

Because Jazz is integrated into the IDE and can recognize processes, it can advise tools on how to behave. So, any tools built on top of Jazz will understand how the team works. And because Jazz is an integrated platform, it can collect information about a project as that project progresses. This means developers can get that information without having to ask for it.

By integrating the collaborative toolset into the IDE, the project also aims to provide multiple collaboration features. For example, the current set includes classic asynchronous collaboration for work items, synchronous collaboration in the context of the work, and RSS feeds for notification of changes, approvals, reviews of changes, and so on.

Here’s a snapshot of how Jazz might work:

1. A build breaks owing to a test failure.
2. Using Jazz, a developer creates a bug report for the problem.
3. Jazz links the bug report to both the build and the failed test.
4. Jazz assigns the bug report to an appropriate developer.
5. The developer delivers the fix to the team, which codes the change set.
6. Jazz links the change set to the bug report and to the build.

The team can now trace the path from the failed test to the corresponding source changes and build. None of this requires team members to perform any bookkeeping. The team can follow all the events through RSS feeds, and everyone can access information. (For more details on how the Jazz design is meeting its technical challenges, see the sidebar interview with Erich Gamma.)

### Both open and commercial

The Jazz platform rests on open,

standard technologies. Both the client and server are based on Eclipse. The Jazz team server is Eclipse, using text I/O and running on a Web application server (Apache Tomcat or IBM’s WebSphere). It communicates with a relational database (Apache Derby or IBM’s DB2). The client and server communicate through HTTP, Web services, and RSS feeds. The Jazz Web client is Ajax, based on the Dojo toolkit.

Jazz is currently available through the beta 1 version of IBM Rational Team Concert, as well as in several other incubator projects at jazz.net. Access to jazz.net is available to all IBM Rational customers and business partners and to others by invitation. According to IBM Rational’s Vice President of Marketing and Strategy Scott Hebner, IBM has no plans to charge directly for access to jazz.net. “We see value in building a community around Jazz technology at jazz.net. It is key for people to be able to easily participate, use the incubators and betas, and contribute,” he said.

Because Jazz is extensible, everyone from commercial software vendors to open source developers is reportedly welcome to extend it and integrate it with their technologies. The design of Jazz is open in that people can participate in design discussions. They can contribute bug reports and look at source code.

Although Jazz is being developed in an open, transparent way, it’s still very much a commercial software development project. Eventually IBM Rational will build commercial products on top of Jazz. The company expects to ship the first products incorporating Jazz technology in 2008.

IBM expects that third parties, customers, and so on will build other Jazz-based tools and frameworks further downstream. For example, University of British Columbia researchers Shawn Minto and Gail Murphy have already created a Jazz add-on that uses file history to recommend a team member to handle problems that arise.

IBM also anticipates that Jazz will add value to its current product set. In the case of IBM’s ClearCase, for exam-

**Although Jazz is being developed in an open, transparent way, it’s still very much a commercial software development project.**

## An Interview with Erich Gamma

Major design and technical challenges confronting the Jazz project have included the need to build for extensibility, interoperability, and flexibility; to include the right set of collaborative features; and to support transitions between individual and group work. In the following interview, IBM Distinguished Engineer Erich Gamma of IBM Rational Software (Zurich), one of the Jazz project leaders, shares his thoughts on how Jazz is meeting these challenges.

**Extensibility.** Jazz presumably must be extensible enough to include any artifacts generated by collaborative add-ons. If Jazz integrates the IDE (integrated development environment) search and collaborative-artifact search, then file-based search becomes the same as knowledge-based search. How do you ensure that when search is closely integrated with the IDE's core functions, it becomes more contextual and traceable?

**Gamma:** Jazz IDE integration profits from Eclipse's extensibility, for both client-side and server-side extensibility. For example, Jazz contributes a Jazz artifact search to the Eclipse search user interface. To enable artifact searching, the Jazz platform provides an extensible text-indexing service. Extenders contribute an indexer for their particular artifacts—for example, bugs, bug attachments, plans, and wiki pages. Given the different roles and skill sets of people who will need to access information in the Jazz repository, the server extensibility story in-

cludes a Web UI. One can manifest Jazz repository information in a rich client UI, Web UI, or both.

**Interoperability.** Developers might use different vendors' tools, even within the same IDE. They might use a collaborative IDE to interface with members of the same organization or with external teams or customers. Interoperability issues might arise when using instant messaging, email, source control, screen sharing, name directories, or abstract APIs. How do you provide a flexible and extensible API—one not tied to any specific implementation?

**Gamma:** Jazz is building on the Eclipse experience. The goal is to have a platform with well-defined APIs on both the client and the server. For example, when it comes to instant messaging, Jazz provides interoperation with different IM systems (Jabber and Lotus Sametime) through a common API. Jazz also exposes the API, which enables the chat support to implement features such as attaching excerpts from a chat transcript to a Jazz bug report. It's still early and these APIs are evolving. We take API commitments seriously. We will be exposing more APIs over time.

**Flexibility.** Jazz needs to be flexible. Collaborative add-ons to an IDE must be extensible by the developers who use them. How do you ensure collaborative capabilities can accommodate development teams' changing norms, needs, and practices?

**Gamma:** Customization by users is normally done through

ple, Jazz add-ons will provide additional collaborative capabilities.

### Potential limitations

According to the NRC's Singer, the chief constraint that Jazz faces is that it works only on the Eclipse platform. Says Singer, "The only people who can adopt it are those who are using Eclipse."

Singer also feels that some processes might not accommodate Jazz's idea of collaboration. "People use all sorts of tools and ways of communication to coordinate their work, to be able to collaborate, to be able to put together big pieces of software," she says. "Some of this has to do with following a particular process. Where Jazz might be constraining is when the model behind it does not jive with these preexisting processes."

Meanwhile, Mike Milinkovich, the Eclipse Foundation's executive director,

told *eweek.com* last March that IBM developers account for as much as 80 percent of Eclipse's development team. He questioned whether that kind of environment is good for Eclipse or Jazz. He also noted that some have charged IBM with killing off the Jazz developer tool competition with Eclipse. Finally, he wondered whether having two open source communities—one for Jazz and one for Eclipse—will ultimately weaken Eclipse.

Clearly, one problem Jazz developers have faced is finding ways to address the needs of a wide spectrum of end users. Programmers in different parts of the world might have different skills, backgrounds, and expectations. Given these differences, John Miano, who runs his own programming firm in the US, is skeptical that IBM's collaborative tools will make any real contribution to the efficacy of offshore software development. "Tools like [Jazz] address the

superficial problems of offshoring, not the flawed assumptions that have driven it," he says.

A related problem facing the Jazz project is that developers who are comfortable working in traditional Eclipse views and editors might have little interest in working with a collaborative-tool user interface. The challenge is to create a set of interfaces that let individual developers work the most effectively. These interfaces must also accommodate developers' multiple roles. And all the interfaces available to the team must provide everyone with a common understanding of project fundamentals such as dates, requirements, and ownership.

In the end, everyone must be comfortable working with Jazz. The danger is that process guidance will come across as something imposed, coupled with an endless number of error and warning messages.

user preferences. For example, users can customize the Jazz notification support by defining how an event should be delivered to the end user. For developers, the Jazz platform provides defined APIs for extensibility, as I already described.

**Collaborative features.** Selecting the right set of collaborative features might be difficult. Each development group works in its own way, so it will have different collaborative-tool preferences. How do you deal with the fact that one group in an organization might prefer to communicate by chat for some types of information and email for others, and that these preferences might change over time?

**Gamma:** Jazz is what we call process aware—a team can specify the process it wants to follow. The process specification governs the Jazz tools and impacts their behavior. For example, a team can define how it wants to handle state transitions of bugs, workflows, and who may perform a particular operation. Earlier in the cycle it can allow a fairly loose process, and at the end of the cycle a much more rigorous process can require reviews and approvals for all code changes. The same applies to the steps the team must take when sharing code. What's important to us is that a team owns its process and that the team can refine the process for its needs.

**Transition support.** Collaboration might involve individual and group activities. Individual developers might prefer to

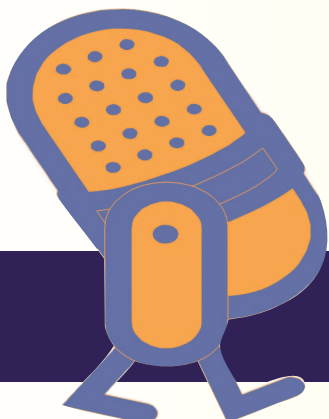
work in their own private sandbox and remain apart from the outside world of the team until they submit their work for inclusion in the integration milestone. How does the tool know when and how to make the transition from individual to collaborative activity?

**Gamma:** This distinction is important in Jazz, and it shows up in different areas. The obvious one is when it comes to source code changes. The Jazz source code management system supports personal workspaces and provides flexible support to share changes using what are called *streams*. Because personal workspaces are managed on the server, you can selectively share your changes with another team member without having to share them with the entire team. The personal/team distinction also shows up when it comes to events from the Jazz server. In Jazz these events are delivered as RSS feeds. Users can subscribe to feeds that contain events about themselves only, or they can subscribe to events for the entire team. Similarly, in the planning support, users can switch between a personal view of the plan that shows only their items and a shared view showing the work for all the team members. While support for the private sandbox is important, at the same time we want to increase transparency within the team. The idea is that you can find out what's going on with other team members without having to ask. For example, a developer can be aware of changes other team members are making, allowing him or her to spot potential conflicts early.

Will Jazz gain widespread acceptance? Says Janice Singer, “Different companies have different cultures for coordinating and communicating in their work. They have models

for how this is going to work. Whether Jazz will be [successful] or not depends on whether the model of collaboration that is engendered in Jazz maps to the models of collaboration or cultures of

collaboration in particular companies. If it doesn't, [success will depend on] whether or not those companies are willing to, or can, change their models of collaboration.”



# Software Engineering Radio



The Podcast for Professional Software Developers  
every 10 days a new tutorial or interview episode

**se-radio.net**