Jazz as a research platform: experience from the Software Development Governance Group at IBM Research

Ping Cheng[■], Sunita Chulani[♦], Ya Bin Dang[■], Robert M. Delmonico[♦], Yael Dubinsky[●], Kate Ehrlich[♦], Mary E. Helander[♦], Tim Klinger[♦], Alexander Kofman[●], Paul M. Matchen[♦], V.T. Rajan[♦], Gilad Saadoun[●], Andrew Sempere[♦], Peri L. Tarr[♦], Clay Williams[♦], Pei Feng Xiang[■], Avi Yaeli[●], Shun Xiang Yang[■], Annie T.T. Ying[♦ *]

[■] IBM China Research Center [●] IBM Haifa Research Center [♦] IBM Watson Research Center

[*] Contact author. Email: aying@us.ibm.com

ABSTRACT

Does the Rational Jazz® platform provide adequate services to facilitate the creation of research prototypes in an end-to-end, full lifecycle domain like software development governance? Jazz is a platform for seamlessly integrating development activities, artifacts, and teams throughout the software lifecycle. The Software Development Governance group—whose vision is to help businesses and IT organizations understand and increase the delivered value of software while managing the risk—has been using Jazz to implement research prototypes that promote developer communications, management of decision rights and decision points during software development processes, and the scheduling of software projects. We also used Rational Team Concert® (RTC)—an Eclipse-based development platform built on Jazz—to aid our own globally distributed development efforts. This paper describes our experiences extending and using Jazz and RTC. It also provides some recommendations for other researchers considering extending the Jazz platform.

Categories and Subject Descriptors

D.2.9 [**Management**]: Software Process Model, Productivity, Lifecycle, and Programming Teams. D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement.

General Terms

Management, Measurement

Keywords

Management, Software Development Governance, Processes, Jazz, Social Network, Decision Right, RACI, Scheduling

1. INTRODUCTION

The Software Development Governance[1] (SDG) group at IBM Research is a globally distributed team of researchers working at multiple locations in the United States, China, and Israel on a set of fundamental capabilities required to support *software development governance*. The need for governance is strong and growing as businesses today face a variety of pressures, including global competition, increasing customer demands, and legislative and regulatory requirements for increased accountability and compliance. Most businesses rely on software technologies to help them cope with these pressures and to differentiate them from their competitors.

As software has become more central to businesses, two important needs have emerged. First, businesses require techniques for understanding and managing the *value* provided by software development and delivery. This is crucial to help businesses optimize their investment in software-related activities. Second, businesses need insight into the *risks* incurred through software development and delivery activities, at both a technical and business level. These risks include cost overruns, schedule slippage, quality issues, and failure to understand and deliver the functionality the business needs most.

Our team's goal is to create and experiment with novel governance solutions to identifying and managing risk and value in software development. We have prototyped three of our current projects with Rational's new Jazz[2] platform:

- **Ensemble** helps identify and remediate communication failures among developers doing related work. It is built using Jazz's team support.
- **Governor** aids IT organizations to identify and manage key decision points during software development processes, and to enable the assignment and exercising of decision rights. It extends the process capabilities provided in Jazz.
- **Tempo** assists developers and project managers in understanding and managing the variability in project scheduling, which is a key source of risk. It is built on Jazz's planning capabilities.

Moreover, we also used Rational Team Concert (RTC)—an Eclipse-based development platform built on Jazz—to support our globally distributed development activities.

In this paper, we describe how the Jazz platform and RTC both helped and impeded the design and implementation of these three prototypes. We also provide some recommendations for other researchers considering extending Jazz.

[1] http://www.softdevgov.net
[2] http://www.jazz.net

2. JAZZ: THE GOOD

SDG is a full-lifecycle set of activities that involves people, processes, and artifacts. Jazz is a platform for seamlessly integrating development activities, artifacts, and teams throughout the software lifecycle. Hence, our goals and requirements aligned readily with Jazz, and we quickly discovered that many capabilities provided by Jazz would become foundations for these three projects. In the remainder of this section, we discuss specific aspects of Jazz that promoted the development each of the prototypes.

2.1 Ensemble

Ensemble helps developers identify the right people to collaborate with, the right times to collaborate with them, and to stay coordinated with them over time. It leverages several key features of Jazz:

People as first-class entities: Ensemble's most fundamental goal is to improve communication between *people*; therefore, Ensemble requires the ability to represent people, their roles and contributions, and their interrelationships. Jazz defines a The Contributor interface to represent its users (people) as first-class entities in the Jazz environment. Contributors have profiles and can be associated with different roles on multiple projects and teams. This interface has, therefore, been particularly useful in implementing those parts of Ensemble that pertain to identifying potential collaborators. Facilitating their communication is simplified by Jazz's built-in integration with chat programs, such as IBM Lotus Sametime[3]. This kind of integration is attempted by existing work (e.g., [2][1]) without Jazz but requires substantial work.

Work Items and traceability: Ensemble analyzes each developer's past and current work to identify others who are working on related artifacts and tasks as potential candidates for communication [4]. The analysis currently examines artifacts (notably, source code), *Work Items* (Jazz's Bugzilla[4]-like task report), and *change sets* (a semantically related set of changes on files delivered to the Jazz repository). Work Items provide useful context information for Ensemble's analyses, such as task description, owner, state of the task (e.g., new, in progress, resolved, etc.), related tasks, other contributors who have expressed interest in this task, and comments that various developers have contributed towards understanding and resolving the task.

These features are analogous to the context provided by bug reports in Bugzilla. However, because Jazz—aided by its RTC development environment—provides integrated support for the full software lifecycle, it can offer additional and fully automated traceability through Work Items, such as traceability from change sets to Work Items, and Ensemble relies heavily on traceability and other relationships to provide good communication recommendations. Bugzilla, as a standalone tool, cannot provide the same degree of automated traceability—analysis of Bugzilla bug reports to infer these traceability links would require heuristics, which take more work and result in less accuracy. Another benefit provided by Jazz and RTC is change sets. Other similar work (e.g., [3]) based on other source control systems need to infer the change sets using heuristics.

Change events and RSS feeds: To help developers stay connected once they decide to coordinate, Ensemble keeps track of what new work each developer's collaboration partners is doing and makes this information available to the developers. Therefore, it requires filterable change notification on Work Items, artifacts, and change sets. Jazz provides this in the form of a service that renders Jazz change events into Atom or RSS feeds.

Seamless integration with the IDE: A key implementation goal for Ensemble is to provide its support as seamlessly and non-intrusively as possible, so as not to interfere with developers' familiar metaphors and workflows. Hence, Ensemble is integrated with RTC, an extension of Eclipse and Jazz's first IDE client. Leveraging the Eclipse-based RTC, Ensemble's support is provided within the same context where developers work. For example, developers can follow links to the artifacts their collaborators are modifying and examine them in an editor without changing context.

2.2 Governor

Governor helps developers follow their team's and organization's chosen best practices and policies, assists decision-makers in determining when it is time for them to act, and aids management in understanding and measuring ongoing development activities. Some of the requirements on a platform suitable for implementing Governor is presented in a paper by Yaeli and Klinger [5]. More specifically, Governor benefited from several Jazz features:

Integrated, end-to-end support: Governor requires "hooks" into development activities and processes, the tools used by developers, and the different artifacts. Jazz provides integration across repository (artifacts), tools, activities and processes, which makes it exceptionally well-suited for Governor's needs.

Governance Solutions via Process Configuration in Project/Team Scope: Governor supports the specification of governance solutions based on an *artifact state transition model*: the movement of artifacts from one "interesting" state to another may indicate a relevant change in process state, and at these transitions, the team, management, or organization may impose requirements to be addressed (e.g., ensuring that an architect approves any interface-level changes before they can be submitted for testing). Jazz provides a configurable process framework that enables control over artifacts and how they are manipulated, and it associates process specifications with projects and teams. Therefore, Jazz enabled us to enact governance solutions for individual teams by configuring the team's process specification. Governor implements enactment of governance specification by configuring the Jazz process.

Process advisors and participants: Jazz defines the *operation* concept, which is an action provided by a component to enable the manipulation of its types of artifacts. "When an operation is invoked, components can provide an opportunity for the governing process to insert appropriate logic both before and after the operation."[5] These additional logic contributions to the operation are called *advisors* and *participants*. Advisors execute before an operation and determine whether or not the operation should run; participants run after the operation. Governor uses

[3] http://www.ibm.com/software/lotus/sametime
[4] http://www.bugzilla.org/
[5] https://jazz.net/learn/LearnItem.jsp?href=content/docs/platform-overview/index.html

advisors to ensure that approvals based on RACI[6] specifications have been signed off. Governor also uses advisors to intercept user operations and implement governance controls. When key decision points are reached, Governor notifies the appropriate people that they need to take action. Governor audits the actions taken and decisions made during enactment for evaluation and subsequent process improvement.

Roles: The definition of roles is key to assigning and carrying out responsibilities. Governor leverages the Jazz Configuration section of each team area to realize the roles specified in that team's governance solution. The governance team sets up roles, responsibilities, and decision rights for every decision associated with a relevant artifact state transition.

Change events and RSS feeds: Governor uses Jazz events in two different ways. First, in response to changes to relevant artifacts, it provides governance decision notification and automatic transition triggered upon a decision. Second, to communicate governance decision outcomes among stakeholders, it creates Jazz change events, which are then communicated via the built-in RSS feed service.

Work Item approvals: Jazz comes equipped with an approval mechanism for work items that enables designated team members to approve or reject the work done. Governor leverages this mechanism to implement RACI-based decision-making processes.

2.3 Tempo

Tempo addresses risks to projects caused by schedule slippage. It provides novel capabilities in project estimation, scheduling, and prediction. Tempo computes the probability that tasks, iterations, milestones, and projects will finish on time, given estimates that developers provide. The Tempo prototype is built on the Jazz platform and benefited from the following Jazz capabilities:

Work Item extensibility: Tempo allows developers to specify *best case*, *worst case*, and *expected case* estimates for each Work Item. In contrast with the traditional single-value task estimate, as in Jazz's Work Item and other earlier work (e.g., [1]), Tempo's approach allows developers to express their confidence in, or uncertainty about, each task. Jazz supports this with a mechanism to extend the fields of a Work Item (by modifying the process specification) so that Tempo can include these additional estimates.

Tasks, iterations, milestones, and projects as first-class entities: Tempo helps project managers understand the probability that work in different granularities—*tasks*, *iteration*, *milestone*, and the *project* as a whole—can accomplish its objectives by a given date. To do so, Tempo requires various inputs these units of work, such as the person assigned to the work, the expected deadline, and the work units' interrelationships. In Jazz, tasks, iterations, milestones, and projects are first class entities. Jazz provides interfaces that represent these units of work. These interfaces made the retrieval of the necessary inputs easy. Project scheduling such as Microsoft Project and FogBugz[7] cannot take advantage of this Jazz's support.

[6] **RACI = R**esponsible, **A**ccountable, **C**onsulted, **I**nformed. It is a means of assigning roles and responsibilities. See http://en.wikipedia.org/wiki/RACI_diagram.

[7] http://www.fogcreek.com/FogBUGZ/

Planning support: Jazz's Agile Planning support aims to help a team schedule the work to be done during an iteration and assigning individual Work Items to members of the team. The focus Jazz has put on planning makes Jazz an ideal environment to provide many of Tempo's support for managers to identify risk factors. For example, Tempo provides insight into when the project is most likely to finish, when it is possible to change the product delivery date. When the delivery dates are firm, Tempo reveals which tasks are not likely to be completed on time, and so the project manager can defer those tasks to later iterations or make tradeoffs among them in the current iteration. In addition, Tempo assists project managers in identifying alternative task schedules that improve the probability that the team will finish a set of tasks on time.

Web UI: Tempo helps both developers and project managers. The Eclipse-based RTC is an excellent tool for developers, but project managers do not require an IDE and are better served by simpler tools. Therefore, Tempo leverages Jazz's Web UI support to provide a web-based interface. This is a better solution for a broad range of users, including managers.

3. JAZZ: THE BAD

Jazz provides a wide range of critical capabilities for team collaboration across the software lifecycle, and our experience is that its support enabled us to do many things. However, Jazz is still a new platform, and we encountered a number of issues that impeded our work. These are opportunities that Jazz can be enhanced. We also offer some suggestions on such improvements.

3.1 Process Support (Jazz and RTC)

Limited sharing across process components: Governor's implementation includes a set of Jazz process components that need to share configuration information. We didn't find an easy way to implement this, either due to lack of documentation or a real limitation of the platform, so we ended up duplicating the shared configuration data across multiple components.

Inadequate ability to manipulate process configurations programmatically: Governor needs to modify process configurations programmatically. We could not find public APIs in RTC that permit easy manipulation of the process configuration (e.g., via a DOM-like interface), except via overriding the current XML configuration specification directly.

Problem resolution: Jazz defines a fixed behavior for resolving problems reported by advisors. Some additional flexibility or programmable APIs to allow clients to provide alternate error handling based on the errors that were found would be useful. Finally, we found no support for web client problem resolution.

RSS feeds and events: Ideally, Governor would want to use the RSS feeds and events to notify people when specific decisions are made. However, Jazz's events mechanism does not provide the event filtering needed by Governor. The mechanism issues notification upon *any* events related to a subscribed team, project, or Work Item. Some event filtering support based on event type or attribute value would be useful.

Documentation and tool support for process specification schema: Jazz provides very little support for understanding and changing process specifications. We had a hard time finding the schema of the process specification, full documentation, and comments on the process specification itself. However, we were

encouraged to see the appearance of some minimal documentation this month on the Wiki on Jazz.net.

3.2 Extensibility (Jazz and RTC)

Extensibility of Jazz process specifications: Jazz supports extensions of repository items, such as process-specific custom attributes of Work Items, as we did in the Tempo prototype. However, the extensibility provided by Jazz is very limited. We could not, for example, implement error checking on fields within a process specification, even though conceptually, the advisor mechanism can handle error checking. The advisor mechanism should become a more powerful mechanism for error checking on Jazz repository items.

RTC UI extensibility: We encountered multiple instances where the UI extensibility in RTC was not adequate for our purposes. As one example, we wanted to implement one of the Ensemble widgets in built-in the Team Area page, which is the locus for team-related information such as the team's members and iteration plans. RTC permits the addition of basic widgets, such as fields, to this page via the process specification, but it provides no support for adding a more sophisticated, custom-made widget. As another example, we found the hover widget to be insufficiently extensible to support its reuse in a different context in Ensemble. The hover widget only supports the use of text or HTML segments, but not the richer SWT controls with call-back actions, such a checkbox.

Limitations on defining new Jazz item types: Jazz requires platform extenders who define new item types to separate physically those methods that manipulate data attributes of an object (which can be part of its "common" component) from those that perform computations involving the contents of the repository (which must be defined in its "service" component). Particularly in the case of computed attributes—which are not semantically different from other attributes and so should not be differentiated at the interface level—the forced interface-level separation of an implementation-only concept necessitates an unnatural and inconvenient interface structure.

3.3 Repository Issues (Jazz)

Preparing a demonstration or testing database: Populating Jazz databases with "canned" data was not easy. Doing so programmatically was not trivial, and doing so manually is tedious and required significant manual work. Some programmatic support in this area would be very helpful for prototyping activities, and also to facilitate testing.

Accessing the database directly: To support some exploratory work in the Ensemble project, we tried to mine data directly from a Jazz database itself, but we discovered that Jazz provides little support for this or information about the structure of, and schema for, the more than 200 tables in a Jazz database. The only document we could find that describes the database structure is a document on the data warehouse. However, this document only describes around 30 major database tables relevant to the data warehouse, omitting the large majority we needed to understand.

3.4 API and Documentation Issues

Incompatibility of client and server APIs: Our first Tempo prototype was implemented as a plug-in for the Jazz client. We eventually realized that we needed to provide web support as well. Surprisingly, this necessitated a very significant revision. Any Jazz extension based on the Web UI component requires code that uses the server API. However, the client and server APIs are completely different. We ended up spending substantial time learning the server API, even though we already implemented a client plug-in.

Documentation: In these early stages of the software, Jazz focused on documenting general usage of the system. While enhancing the system is encouraged, API support and documentation was very limited. The lack of documented Eclipse extension points really hindered our productivity. It forced us to spend lots of time reading source code in order to integrate our enhancements.

One important lesson we learned was that before you start reading the code, you should go to Jazz.net's Wiki and forums. These are loaded with useful information. Moreover, other developers (as well as members of the Jazz team) frequent the forums, and they may be able to point you in the right direction.

4. CONCLUSION

We chose the Jazz platform to implement three novel SDG research prototypes. These prototypes leveraged several major Jazz capabilities: collaboration, processes, events, and planning. We also identified some opportunities for improving the Jazz environment to make it more amenable to researchers in software development. Overall, we found the Jazz platform to be valuable, and we recommend it to others. However, while RTC is available as a 1.0 product, the underlying Jazz platform API's are still at beta level and they are not well documented. As a result, there are still significant challenges in extending the platform.

5. ACKNOWLEDGMENTS

We thank Murray Cantor and Ben Waber for the discussion and help on this work. IBM, Jazz, Rational, and Team Concert are trademarks of IBM Corporation, in the United States, other countries or both.

6. REFERENCES

[1] Boehm, B. Software Engineering Economics, Prentice Hall, 1981.

[2] Fitzpatrick, G., Marshall, P., and Phillips, A. CVS integration with notification and chat: lightweight software team collaboration. In Proc. of the Conf. on Computer Supported Cooperative Work, 2006.

[3] Minto, S., Murphy, G. C. Recommending Emergent Teams. In Proc. of the Int'l Workshop on Mining Software Repositories, 2007.

[4] Valetto, G., Helander, M., Ehrlich, K., Chulani, S., Wegman, M., and Williams, C. Using Software Repositories to Investigate Socio-technical Congruence in Development Projects. In Proc. of the Int'l Workshop on Mining Software Repositories, 2007.

[5] Yaeli, A. and Klinger, T. Enacting responsibility assignment in software development environments. In Proc. of the Int'l Workshop on Software Development Governance, 2008.