Software development
White paper
August 2007

Rational. software

Introducing collaborative development environments.

Grady Booch, chief scientist, IBM Rational

Contents

Introduction

In this closer look at collaborative development environments (CDEs) Grady Booch, the IBM resident expert on CDEs, describes what they're all about, provides real-world examples and explains why this emerging technology is important and relevant to the developer and technical communities.

Collaboration and the emergence of CDEs

A CDE is a virtual space where the stakeholders of a project can brainstorm, discuss, deliberate, negotiate, record and generally labor over ideas and information—even if they're separated by time or space—to carry out some task, most often to create a useful artifact and its supporting objects.

Collaboration is an element of every engineering domain. As described by Coplien in "Organizational Patterns of Agile Software Development,"* there are some well-defined and well-understood patterns of behavior in hyperproductive teams. Opportunities to encourage and amplify these patterns are available through the use of a CDE. The rise of the Web as a natural extension of an individual's physical and daily world; the economic pull of outsourcing and offshore development; the integration of third-party software; the emergence of Web-centric service-oriented architectures; the use of home and remote offices; and the growth of strategic partnerships among organizations and companies have all contributed to the increasing distribution of teams, but those teams still need to be able to effectively work together.

Collaboration has always been an essential part of the fabric of the Internet: e-mail, instant messaging, chat rooms, discussion groups and wikis are common collaborative elements that have matured over time; so in one regard, there is nothing new here. Collaboration among teams is already facilitated through the use of features embedded in standard desktop products such as office suites, where there is ample support for shared document reviews, distribution of documents among teams and mechanisms for performing common collaborative tasks. In a practical sense, these tools provide the baseline of collaboration functionality for today's software developers.

However, there are two elements that make CDEs for software development materially different: First, software developers must manipulate semantically deep artifacts with equally semantically deep associations among them. (This is in contrast to the kinds of semantically weak textual documents commonly manipulated by business organizations.) Second, the Web is essentially the atmosphere in which all software developers virtually live, and it is only a short distance from physically colocated teams to virtually colocated ones. So teams can leverage the plumbing of the Web and populate this atmosphere with creature comforts that facilitate distributed collaboration.

A CDE is not a unique or special thing. There are hundreds of small things that already exist and that can be combined to form a virtual space. But a CDE is a particularly fragile thing because it touches on the social elements of development, and it's sensitive to issues of presentation, simplicity, ease of use, personalization and culture. A CDE must be lightweight, nimble and snappy in appearance, yet it must be deep in its inner workings. It must never,

ever get in the way of work; rather, it must disappear in the ether, providing a sanctuary in which the individual developer works and in which the team assembles. Thus, the purpose of a CDE is to create a frictionless surface for development by eliminating or automating many of the daily, noncreative activities of the individual and the team, and by providing mechanisms that encourage creative, healthy and high-bandwidth modes of communication among a project's stakeholders.

The changing nature of teams

Every engineering activity involves producing a solution that balances the tangible and intangible forces that weigh upon a system. In software, these forces include the usual business pains (cost, schedule, mission), environmental pains (resources, compatibility, complexity), developmental pains (production, resilience) and operational pains (functionality, performance, quality, dependability), as well issues of value (legality, ethicality, morality).

For systems of any reasonable complexity, no one person can efficiently counteract these forces. For most software-intensive systems under construction, operation or revision, software development is a team sport. In these circumstances, inter- and intrateam interaction, communication and dynamics play as minor a role as individual heroics in successful software development. For this reason, understanding more about optimizing software development team performance is a critical task of software engineering.

There are few hard studies that tell us the median size of a contemporary development team, but our experience across a broad range of domains tells us that teams of four to eight people are most common, with teams of one or two being the second most common. Beyond the range of four to eight members, the

existence of larger teams tends to tail off; although, we also find a peak in the curve of team size and occurrence somewhere around the 100 to 200 mark. Some systems, such as telephony, financial, and command and control, are so complex that they require large teams to complete the work on time.

Not only must an organization focus on the efficiency of each individual team, it must also be concerned about the efficiency of its *teams* of teams. In even a modest-size development organization, there might be 100 or so developers organized in teams of 4 to 8, with most focused on point products and a few focused on infrastructure technologies that support all the other teams. In practice, most of these individual teams will themselves be contiguous (that is, their cubicles will be physically close to one another), which encourages the jelling of the team through the myriad informal interactions that occur during the day. However, relative to one another, these teams will typically be physically disconnected, thereby reducing the level of informal contact as well as the bandwidth and quality of interteam communication. Indeed, one of the problems any such organization faces is simply keeping these teams of teams on the same page. That requires sharing project status, reducing duplication of work, engineering the architectural seams among groups, and sharing knowledge and experience.

In short, delivering better software faster involves the rhythms of the individual developer, the small team of developers and—for larger projects—teams of teams, and even teams of teams of teams of developers.

The difference between IDEs and CDEs

Software developers spend a majority of their time on code-centric activities supported by an integrated development environment (IDE) that offers a range of code development and manipulation features. Other aspects of their work

that involve interaction, communication and coordination within and across teams are generally supported by a discrete combination of capabilities, such as configuration management systems, issue tracking databases, instant messaging (IM) systems and project Web sites. Assembled in a coherent fashion, this latter set of capabilities can compose a CDE for software engineers.

Whereas traditional IDEs focus on improving the efficiencies of the individual developer, CDEs focus on improving the efficiencies of the development team as a whole. While it is the case that most modern IDEs have some degree of collaborative support (for example, the ability to check in and check out an artifact from a common repository or to call out to Microsoft® NetMeeting software and whiteboards from a menu), an IDE will not transmogrify into a CDE if you just incrementally add collaborative features. IDEs are essentially developer centric, meaning that their primary user experience focuses on the individual developer. CDEs are essentially team centric, meaning that their primary user experience focuses on the needs of the team (but with points of entry for different individuals). Psychologically, this is a subtle yet important shift of perspective.

To apply the thinking of Abraham Lincoln, software development takes place of the Web, by the Web and for the Web. There is considerable continuing development of the Web's infrastructure; similarly, a great deal of application software is being developed for Web-centric systems. Relative to CDEs, however, development by the Web means using the Web to change the nature of software development itself.

CDEs as Web based, artifact centric and multidimensional

The most common IDEs (Microsoft Visual Studio and the open source Eclipse) are thick-client centric; they cater to the code warrior and provide a single, complex window whereby the programmer peers into the system under construction. But emerging CDEs are and should be Web based, artifact centric and multidimensional. The Web is an ideal platform for doing software engineering because it permits the creation of virtual spaces that transcend the physical boundaries of its participants. CDEs should also be artifact centric, meaning that they should offer a user experience that makes work products primary and engages tools only as necessary. Finally, a CDE should be multidimensional in the sense that different stakeholders should be offered different views (some via browsers, some via thick clients), each adapted to that stakeholder's specific needs.

Collaborative sites both on and off the Web have existed for some time, but we began to see collaborative sites focused solely on software development starting about 10 years ago. Most of these sites were neither public nor reusable, but rather one-off creations of specific projects. One of the earliest such sites was for a large command and control system being built by Boeing. (As we will discuss later, it should come as no surprise that Boeing has continued to innovate mightily in this space.) As part of an architectural review, viewers were placed in front of a homegrown intranet that contained every artifact created by the project, from vision documents and models to code and executables. Although this site offered little in terms of collaborative mechanisms, it did offer a virtual presence, a veritable electronic meeting place for the project's team members, many of whom were geographically distributed.

Soon after, we saw commercial sites emerge for the construction and computed-aided design (CAD) industries, both using the Web to provide a virtual project space. Similar sites grew up for the open source software development industry. Indeed, since a great deal of open source code is written by individuals who never interact with one another in person but only via e-mail and the Web, it is natural that the Internet be used to provide a sense of presence for open source projects.

The general purpose of a CDE

The purpose of a CDE is to create a frictionless surface for development that eliminates some of the pain points in the daily life of the developer that individually and collectively impact the team's efficiency, such as:

- *The cost of start-up and ongoing working space organization.*
- *Inefficient work product collaboration.*
- *Maintenance of effective group communication, including knowledge and experience, project status and project memory.*
- *Time starvation across multiple tasks.*
- *Stakeholder negotiation.*
- *Stuff that doesn't work.*

We call these "points of friction" because the energy lost in their execution could be directed to more creative activities that contribute directly to the completion of the project's mission. Addressing these points of friction represents substantial return on investment for organizations.

A CDE can address many of these points of friction. Making a virtual project environment just a URL away can minimize start-up costs; being able to self-administer such sites also means that the team can manage its own artifacts rather than require the services of a dedicated support team. The friction associated with work product collaboration can be minimized by offering artifact storage with integrated change management and the storage of metaknowledge. Communication can be facilitated by the use of mechanisms for discussions, virtual meetings and project dashboards. Time starvation can be addressed not only by a hundred small creature comforts for the developer, but by making possible virtual agents that act as nonhuman members of the development team who are responsible for carrying out scripted, tedious tasks. Stakeholder negotiation can be facilitated by mechanisms that automate workflow. As for stuff that doesn't work, well, a CDE won't make much of a difference: the best we can suggest is that you simply refuse to buy or use products of inferior quality. That notwithstanding, if stuff doesn't work for you, then it is likely that there are others in the world who have experienced the same problem and might have solved it or found a workaround. In the presence of an extended community of developers, such as those you might interact with in a CDE, mechanisms for sharing experiences can temper the problems of hard failure (and perhaps offer a form of collective bargaining to put pressure on the vendor who has delivered a shoddy product).

The software development purpose of a CDE

There exist only a few commercial CDEs focused primarily on the problem of software development over the Web (most notably, SourceForge and Collab.net), but there are many more that have been created for other domains or that address one specific element of the software CDE domain. Studying these different sites can help us understand what a CDE is, what it is not and what it can be.

The construction, manufacturing and electronics industries have been fruitful places for the evolution of collaboration products. In fact, these industries have been the earliest adopters of collaborative technology, perhaps because many of their points of friction are directly addressed by the features of a CDE. This domain is so well suited to collaboration products that a trade organization (Network for Construction Collaboration Technology Providers) was established to form standards for this domain.

Imagine, for example, a building being erected in Kuala Lumpur. Onsite, the construction supervisor might encounter a design problem whose resolution would require the interaction of the building's architect, structural engineer and client (and, of course, lawyers). If the architect were in London, the structural engineer in New York and the client in Hong Kong, getting these stakeholders together in real time would be problematic. Instead, using the Web as a virtual meeting place for the project, these three people (and the lawyers) can come to a resolution in real time.

This kind of collaborative development is what lies behind products such as Gehry Technologies' Digital Project tools. Frank Gehry, the architect of dramatic works such as the Guggenheim Museum in Spain, formed Gehry Technologies to commercialize the tools that he used to design and construct his architectural creations. The Digital Project tools include Designer, Foundation, Structures and MEP for modeling; Viewer and Project Manager for project management; and Knowledge Template and Knowledge Adviser for model checking.

Moving to the domain of manufacturing, Boeing, together with IBM Business Partner Dassault, created the Global Collaboration Environment to design and build the Boeing 787 aircraft. This Web-centric system provides modeling, asset management and product management tools, with an emphasis on traceability and auditability. Dassault has its own collaboration tools, most notably ENOVIA for collaboration across the manufacturing chain. The company's SmarTeam Web editor offers tools for the distributed manipulation of 3-D models. Dassualt has even created a robust community of practice for its collaboration products.

CDEs at your fingertips

A CDE is not so much a single killer application as it is a coherent collection of many small things. Although not really CDEs in their own right, some technologies provide critical collaborative infrastructures for full-blown CDEs. Of these, IM and e-mail are probably the most pervasive mechanisms for collaboration. Products such as NetMeeting are also commonly used by teams for ad hoc conferencing (with WebEx and its equivalents providing more scalable facilities). Although subtly different in their user experience, both services offer Web-centric conferencing with the ability to broadcast documents and slides (for lectures) and share desktops (offering the moral equivalent of an electronic whiteboard).

There really is a spectrum of infrastructure collaborative mechanisms that may be applied to a Web community, each with its own value. Specifically:

- *Web logs (blogs) are useful as mostly one-way informational sites.*
- *Mailing lists are good for small groups with a common purpose, conversations that wax and wane over time, communities that are just getting started, and newsletters and announcements.*
- *Message boards are useful for asking and answering questions, encouraging in-depth conversations, managing high-volume conversations and providing context, history and a sense of place.*

- *Chat rooms are good for holding scheduled events, preparing for and debriefing life events, discussing offline events in real time and casually chatting.*
- *Whiteboards are useful for brainstorming, communicating and discussing.*
- *Net meetings are useful for one-on-one discussions, as well as for group presentations and distributed discussions.*
- *Portals (such as IBM developerWorks®) are useful meeting places for communities of practice. Not only can the primary owner of the domain provide relevant knowledge to users, but users can share their experiences and solutions.*
- *Wikis, which are rather like message boards on steroids, permit a trusted community to figuratively meet at a water cooler and leave behind the bread crumbs of its members' conversations.*

Most of these infrastructure services are quickly becoming commodities, meaning that they are already available from a variety of sources—in particular, open source.

Conclusion

Effective teamwork is an essential part of every nontrivial software engineering effort. Collaborative capabilities are essential to support these teams, particularly as team sizes get smaller while team interaction becomes more geographically dispersed. CDEs are an ideal way to get people working together effectively, and you probably already use one—you just aren't using it to its capacity.

For more information

To learn more about IBM products and services that support CDEs, contact your IBM representative, or visit:

ibm.com/developerworks/integrate/collaborate.html

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

Produced in the United States of America
08-07

---

* Coplien, James O. and Harrison, Neil B., "Organizational Patterns of Agile Software Development," Prentice Hall PTR, July 16, 2004.

RAW11045-USEN-00