The Future of Collaborative Software Engineering

Jim Whitehead
Univ. of California, Santa Cruz

Acknowledgements: Ivan Mistrik, John Grundy, Andre van der Hoek, Arie van Deursen, Jesus Gonzalez-Barahona
Why collaborate?

At high levels of abstraction, we are slow and error-prone.
So, we must work together to create large projects.
It is human limitations that drive the need to collaborate.
(But we are social creatures, and enjoy this.)
Problems of Working Together

Creation of shared meaning
Organizing people for joint work
Software is thought-stuff, the highly malleable conversion of abstractions, algorithms, and ideas into tangible running code.

A software system is shaped by the intersecting activities and perspectives of the engineers working on it.

As they seek to understand the system, software engineers give it shape.
Collaboration: Quest for shared meaning

Somehow, through the imperfect instrument of language, the vast pool of variable outcomes inherent in any software system must be reduced to a coherent unity.

This, then, is the challenge of software engineering collaboration.

Software engineering collaboration is:

The mediation of the multiple conflicting mental conceptions held by human software engineers.
Model-based collaboration

Collaboration to create shared meaning leads to model-based collaboration

capture system goals and behavior in models
creation of shared meaning around these models
elimination of error and ambiguity within the models

Stakeholders/ Users/Customers

Requirements Engineer

System Architect

Developer

Project management

Software Engineers

Models

Requirements
Use cases, scenarios, requirements statements

Software Architecture
Feature model, arch. diagrams, tradeoff analyses, arch. document

Design
UML, design documents

Code Unit Test Inspection
Source code, test code, build scripts, test harness

Bug reporting, tracking, & fixing
Bug reports, code patches

Collaboration Goals

- Develop shared mental model of system
- Negotiate scope and capabilities
- Elicit requirements from stakeholders

- Drive convergence to single architecture
- Negotiate modular decomposition
- Reduce dependencies among org. units

- Resolve inconsistencies
- Negotiate interfaces and use protocols
- Record design rationale

- Negotiate dependencies among classes and methods
- Identify and eliminate bugs
- Resolve edit conflicts

- Identify and confirm bugs
- Verify and apply code patches
- Identify new feature requests
Future Directions for Collaboration in SE

Advanced development environments
- Improving access to artifacts

Social networking for software engineering
- Tightly knit communities

Broadening participation in software projects
- Collaboration outside development organization

Capturing design rationale as an argumentation structure
- Structure for recording support for shared meaning
Advanced Development Environments
### Centralization and Decentralization

There are forces at play that favor both centralization and decentralization of development environments.

<table>
<thead>
<tr>
<th>Centralization</th>
<th>Decentralization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project artifacts all held in one location (Forge)</td>
<td>Project artifacts distributed across multiple developer machines</td>
</tr>
<tr>
<td>Must be online to access</td>
<td>Permits offline access</td>
</tr>
<tr>
<td>Favors organizations over individual developers</td>
<td>Favors individual developers over organizations</td>
</tr>
</tbody>
</table>
Recent Development Environment Trends

Centralized

Forges become more sophisticated
   Improved metrics, management functions, IDE plugins, translation
Proliferation of forges
   SourceForge, Googe Code, Launchpad, Github, Tigris
Web 2.0 brings interactive applications to the browser
   Gliffy, eRequirements, Google Docs, Bespin

Decentralized

Distributed change management
   Git, Bazaar, Mercurial
Increased modularity in monolithic desktop environments
   IDE plugins
Inclusion of interfaces to network-based services
   Mylyn interface to Bugzilla, Trac, Redmine, JIRA
The Future ... Centralized

Integration with other web-based services
   Especially sites focused on collaboration

End-to-end web-based
   Create/edit all project documents, diagrams, code online

Metric-driven software engineering
   Easy to collect metrics, provide predictions

Centralized control over access to artifacts
   Adding someone to team doesn’t mean they have access to all documents

Increasingly sophisticated software analytics
   Leverage cheap cycles, batch-oriented processing
The Future ... Decentralized

IDEs provide uniform interface across multiple Forges
  Mylyn-style capability, for multiple services
Multi-project, multi-site data integration
  Provide developer-centered view of work across multiple projects on multiple forges
Local work with remote synch (general data replication)
  Distributed SCM style of work for bug reports, discussion lists, etc.
  Analogy: Lotus Domino/Notes for IDEs
Rich software visualizations
  Leverage rich desktop graphics, local code repositories
In reality... both

Likely to see aspects of both the centralized and decentralized future

For collaboration, centralized future is more interesting
Once engineers use a given service, can layer additional coordination services on top

Let’s assume a centralized future... and see where that leads us!
Social Networking for Software Engineering
Rise of Social Networking

Social networking sites now major hubs of social interaction

Facebook: #2 site on web, 29% of global Internet users visit site, 6.2 billion daily page views, 420 million daily visitors. And growing.

Social network as platform drives fast application growth

Zynga, social games company
Farmville: released June 19, 2009
Today: 65+million users
Software Development is not like Facebook...

Facebook is often viewed as a place for maintaining non-work social networks
   A site for fun, not for work
Software development involves joint collaborative work
   Implies different site requirements, use profiles
Some models
   IBM Beehive, Github, Advogato
**Beehive**

Social networking site designed for IBM employees at work

**Uses**

- Connecting with colleagues outside one's group
- Meeting new colleagues
- Caring about others at work
- Networking for career advancement
- Building support for projects

Used by managers, senior level employees, regular employees

Motivations for Social Networking at Work, Joan DiMicco et al., CSCW 2008
Github

A software forge with social network facilities

Public developer profiles

Project/developer news feeds

Weak affordances for social interaction, identity creation.
Advogato

Web of trust ratings for open source developers

Blogging

Few affordances for building social networks beyond trust ratings

Weak connections to forges

advriusa is currently certified at Journeymer level.
Name: Audrius Meškauskas
Member since: 2007-10-05 17:11:50
Last Login: 2009-11-15 06:00:45
Homepage: http://www.linkedin.com/in/audriusa

Notes: I wrote the first working CORBA, RMI, IIOP and SGML-driven HTML parser implementations for GNU Classpath. GNU Classpath is the project where I have first seen CVS, Changelog, real testing suite, release branches and many other things 'live and working'. In the real use I also think I have learned a lot of unwritten rules how the programming is divided and coordinated in a large team, working so differently from the waterfall model - still deadly efficiently. I am currently a software engineer in Spectrasis, a small research-oriented Swiss company, currently coordinating the work of the four developer team (myself including). Spectrasis does not look like a software company from the web site, that is true - but we do have many quite serious projects, some also on Qtopen mobile devices.

I am also studying at ETH university at the moment, seeking the Master of Advances Studies degree (post master studies).

Projects

- Contributor on GNU Classpath
- Contributor on Kaffe OpenVM
- Contributor on Evangelism
- Contributor on GNU ggs
- Contributor on Wikipedia
- Lead Developer on Java applets in Wikipedia

Articles Posted by advriusa

- Dream java: HashSet from Sun, but likely BitSet from GNU 22 Mar 2008 at 21:11 UTC

Recent blog entries by advriusa

Syndication: RSS 2.0

23 Aug 2009 »

Java Applets in Wikipedia - that do you think?
Future of software social networking

Integration of social networking and software forges

Might be best to keep social networks separate, but integrated with forges

Perhaps with integrated statistics for yourself across multiple forges

Provide strong affordances for social interaction

But, keep work focus

Key challenges

Being interesting (avoid “LinkedIn effect”)

Being fun, not silly

Providing value over other sites

Unclear how this will play out

Potential to create portal site used by large fraction of world’s developers

Flickr: erwinkaram
Broadening Project Participation
Broadening Project Participation

Many types of software lock-in their users

Organizations dependent on this software have strong incentive to ensure it meets their needs

Currently:

Customers consulted during requirements...
... then get to give feedback during beta testing

Opportunity:

Develop means to engage customers during all phases
Allow them to actively ensure needs are met

In Rome, lovers attach a padlock to a bridge, and throw the keys into the river Tevere to lock-in their love.

Flickr: James and Vilja
EVE Online

EVE Online is a space-themed online (MMO) game
- Over 300,000 players
- Developed and operated by CCP Games

Council of Stellar Management
- 9 delegates directly elected by other players (8.6%-11% participation)
- Represent interests of players and deal directly with CCP

Face to face summits with CCP
- Council members are flown at CCP expense to meetings
- CCP accepts or rejects proposals provides by Council
- Many proposals have led to new game features

Company viewpoint

“We are not the gods of EVE. We consider ourselves to be janitors or caretakers”
Challenge: What to control

Most organizations do not want to cede total control over software to customers

- Not complete open source

Selective opening of projects

- Choose portions of the code to make available
  - No need to reveal trade secrets, distinctive IP
  - Foreign language translations
- Give customer participants control over some decisions, but not all
  - Or, reserve veto authority

Ideal

- Customers add features and fix bugs most relevant to them, increasing satisfaction
Capturing Project Argumentation
Capturing Project Argumentation

Even experienced engineers disagree on portions of the requirements and design of large systems

“Design for change” means “predict the future”

An argument ensues

Architecture and design can be viewed as argumentative processes

Resolution of differences of prediction and interpretation among skilled practitioners to develop system structure

Since only one model can prevail, architecture and design are simultaneously cooperative and competitive
Capturing Project Argumentation Structure

Need to record why a decision was made
  Primary argument
  Supporting facts
Also important to record alternatives considered, but rejected
  Why another engineer thought it could be done differently
  As assumptions change, these alternatives may become relevant
Argumentation structure gives insight into both the design as-is, as well as design variations
Can computers have free will?

free will: The ability to make voluntary, unconstrained decisions. Freely made decisions are independent of the influence of such deterministic factors as genetics (nature) and conditioning (nurture).

2. Computers can't have free will. Machines only do what they have been designed or programmed to do. They lack free will, but free will is necessary for thought. Therefore, computers can't think.

3. Humans also lack free will. Whether or not computers have free will is irrelevant to the issue of whether machines can think. People can think, and they don't have free will. People are just as deterministic as machines are. So machines may yet be able to think.

4. Nisian Smart, 1964
Humans are programmed. If you accept determinism, then you accept that nature has programmed you to behave in certain ways in certain contexts, even though that programming is subtler than the programming a computer receives.

5. Free will is an illusion of experience. We may think we are free, but that is just an illusion of experience. Actually, we are determined to do what we do by our underlying neural machinery.

6. Philip Johnson-Laird, 1988a
Free will results from a multilevel representational structure. A multilevel representational structure is capable of producing free will. The system must have levels for:
- representing options for action (e.g., go to dinner, read, take a walk);
- representing the grounds for deciding which option to take (e.g., the one that makes me happy, choose by flipping a coin);
- representing a method for deciding which decision-making process to follow (e.g., follow the most "rational" method, follow the fastest method).
Computers that have been programmed with such multilevel structures can exhibit free will.

7. Geoff Simons, 1985
Free will is a decision-making process. Free will is a decision-making process characterized by selection of options, discrimination between clusters of data, and choice between alternatives. Because computers already make such choices, they possess free will.

8. Geoff Simons, 1985
Conditional jumps constitute free will. The ability of a system to perform conditional jumps when confronted with changing information gives it the potential to make free decisions. For example, a computer may or may not "jump" when it interprets the instruction "proceed to address 9759 if the contents of register A are less than 10." The decision making that results from this ability frees the machine from being a mere puppet of the programmer.
Opportunity: Web-based Argumentation

Web-based design argumentation support

- Instead of static arguments on large posters...
  - Dynamic, living argumentation structures
- Explicit support for argument structure
  - Major/minor points
  - Pro/Con arguments
  - Supporting facts
  - Ability to link to (extract out) requirements
  - Activity awareness, hotspots

Analysis of argumentation structure

- Summarization of arguments
- Consistency checking

Support software variability analysis

- Why select a specific feature? Here’s why, and why not.

Some ongoing work in this direction

Compendium [Shum et al. 2005]

Global Sensemaking [www.globalsensemaking.net]
Collaboration in software engineering is a process of mediating different conceptions of a system. Centralized, web-oriented environments of the future provide a framework for rich collaboration services. Examples of such services include:

- Social networking for software engineering
- Broad-based inclusiveness in design and development
- Recording argumentation structures

Overall:

- Increase project awareness and reduce barriers to communication
- Hence opening more opportunities to detect and mediate differing mental conceptions of the system
“The future is here, it’s just not evenly distributed yet.”
Attributed to novelist William Gibson