The **swim** system for use case visualization

Ward Cunningham, AboutUs.org
Bjorn Freeman-Benson, Eclipse Foundation
Karl Matthias, Eclipse Foundaton

Using the computer should feel more like a **conversation** than an **interrogation**.

I created wiki twelve years ago to do with writing what I could already do while programming: exploit the human ability to communicate.

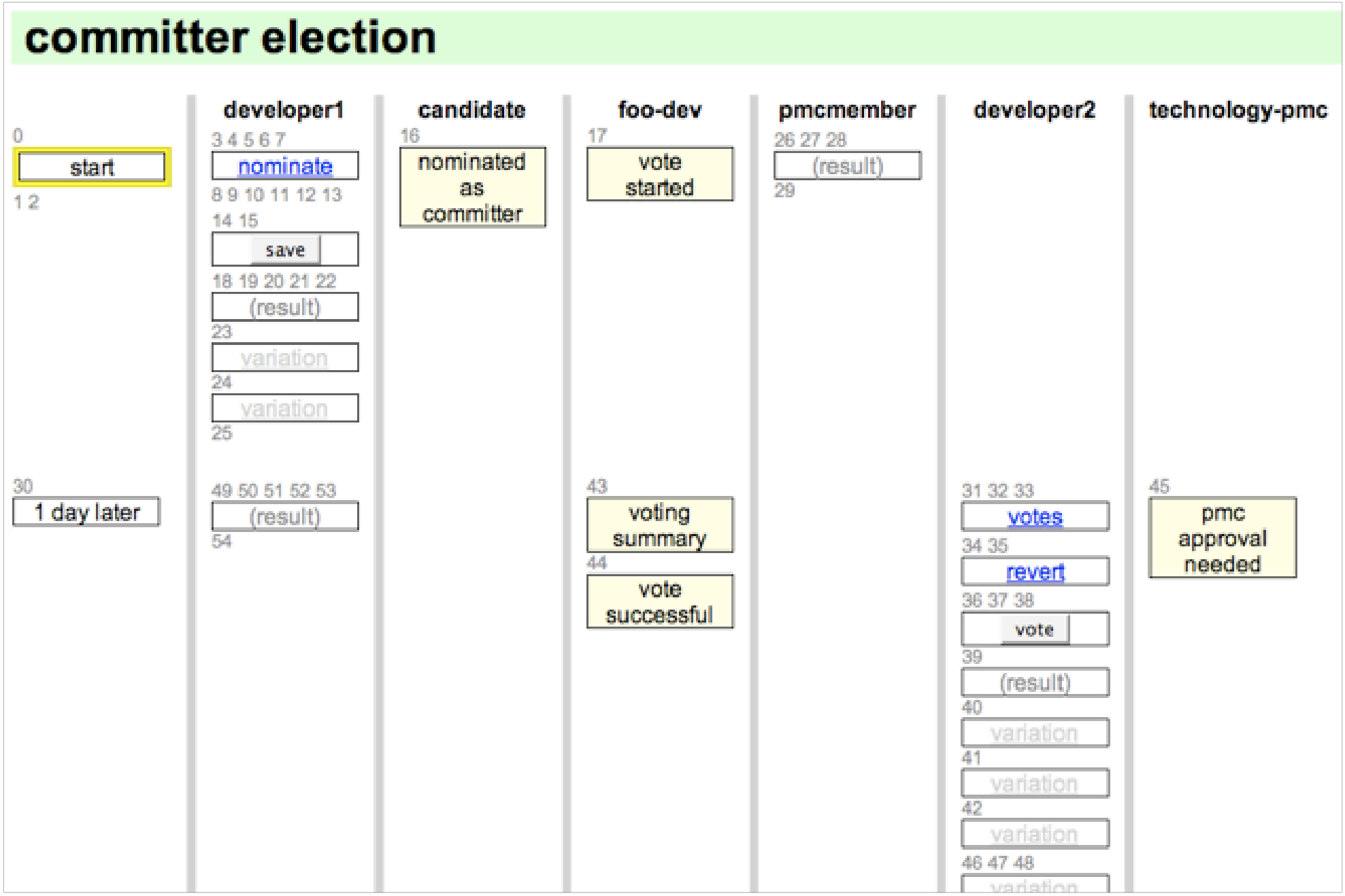
To converse, one must build a model of the beliefs and intentions of the other. In this work I take one step toward the goal above. I expose to the end user all that the programmer knows for sure about what the program will do.

Wikipedians write three times as many sections in discussion of bridge collapse than in article itself.

Extreme Programmers discuss each step of programming as they work.

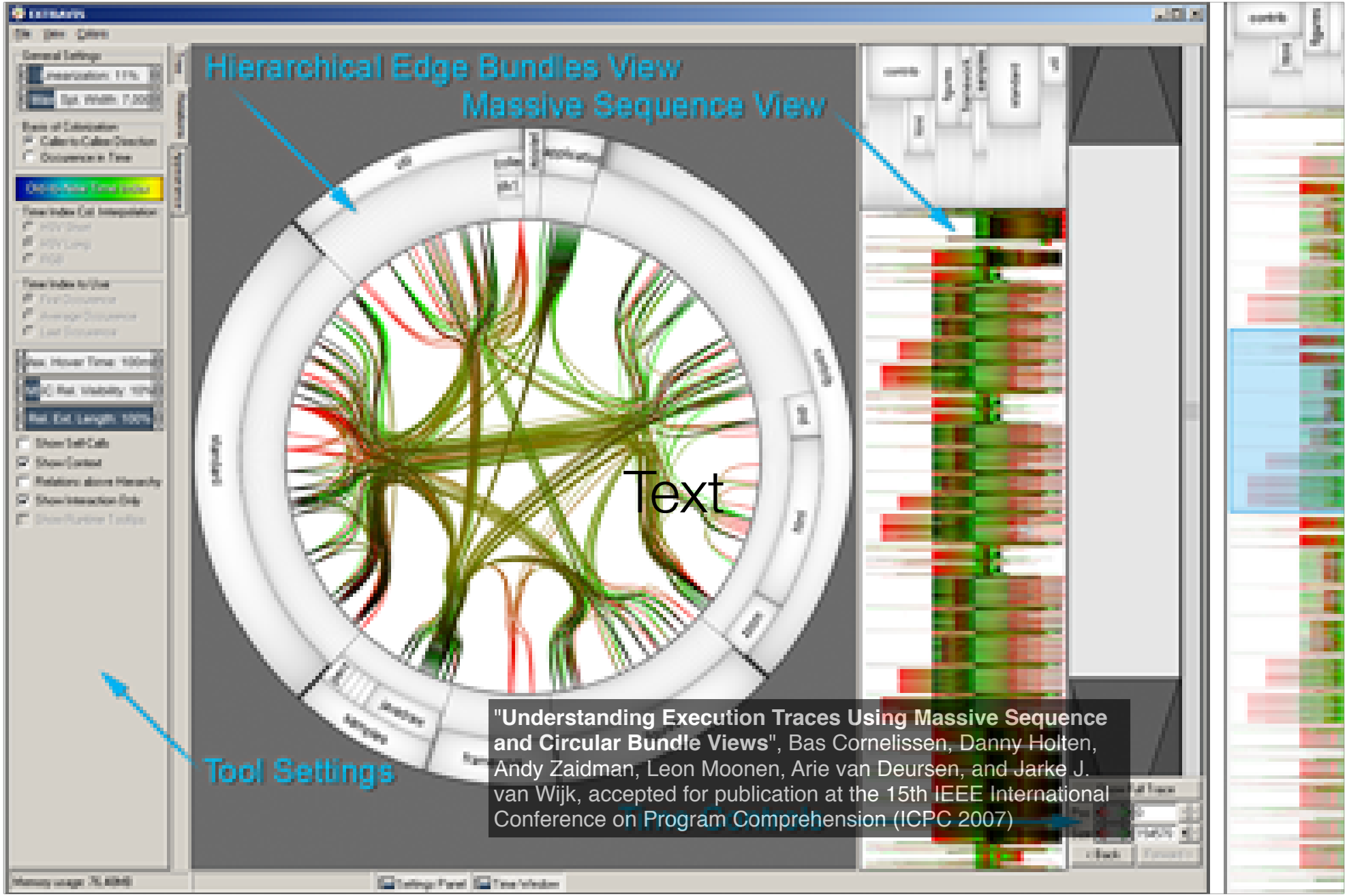
Bjorn and Karl reenact the decision making process that takes place in front of the **swim diagram** on a large computer screen.

Bjorn and Karl reenact the decision making process that takes place in front of the **swim diagram** on a large computer screen.

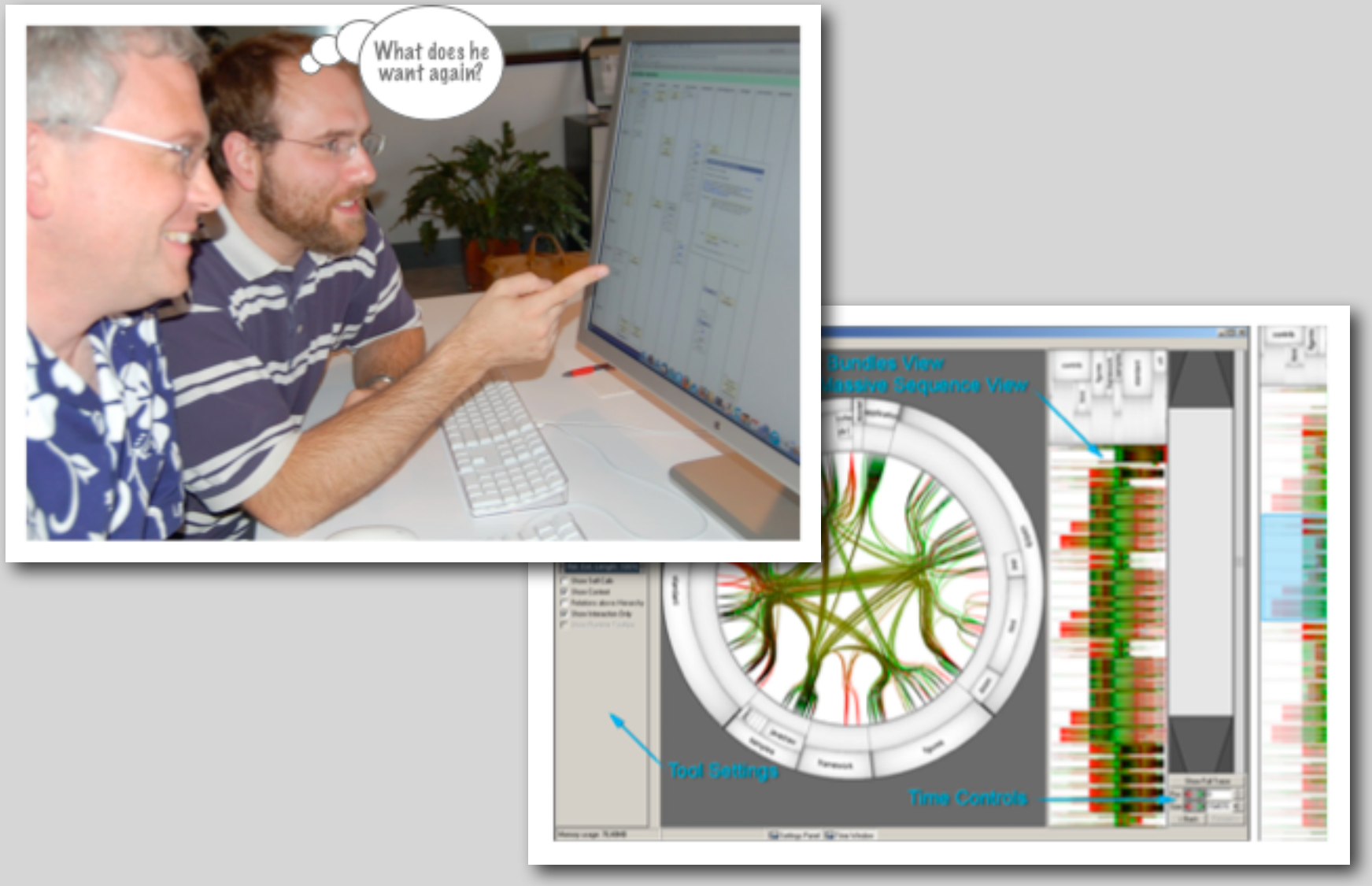


A region of the swim diagram for a committer election showing **where** and **when** things happen.

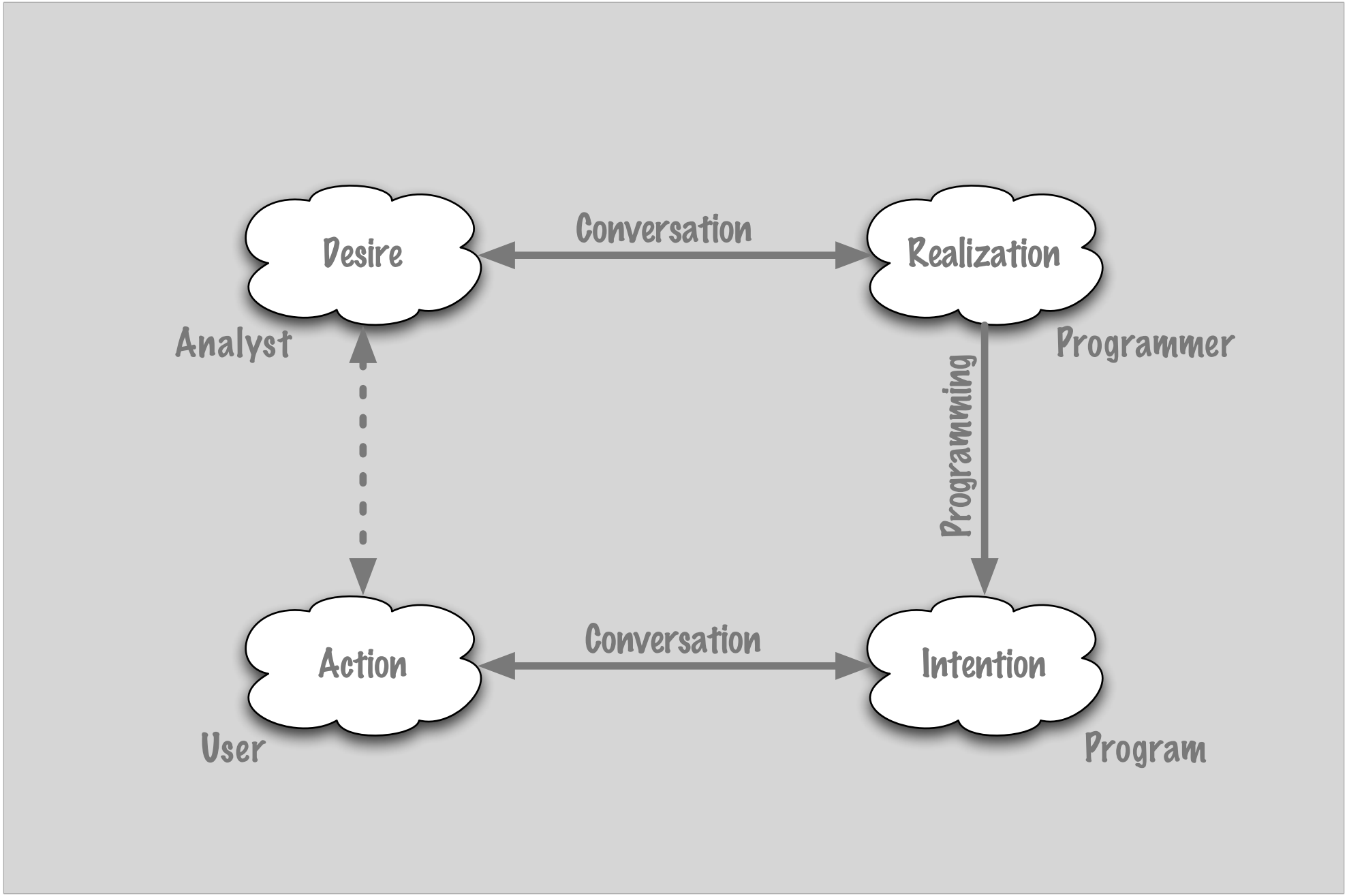
Business process engineers separate roles with bold lines suggesting swimming lanes.



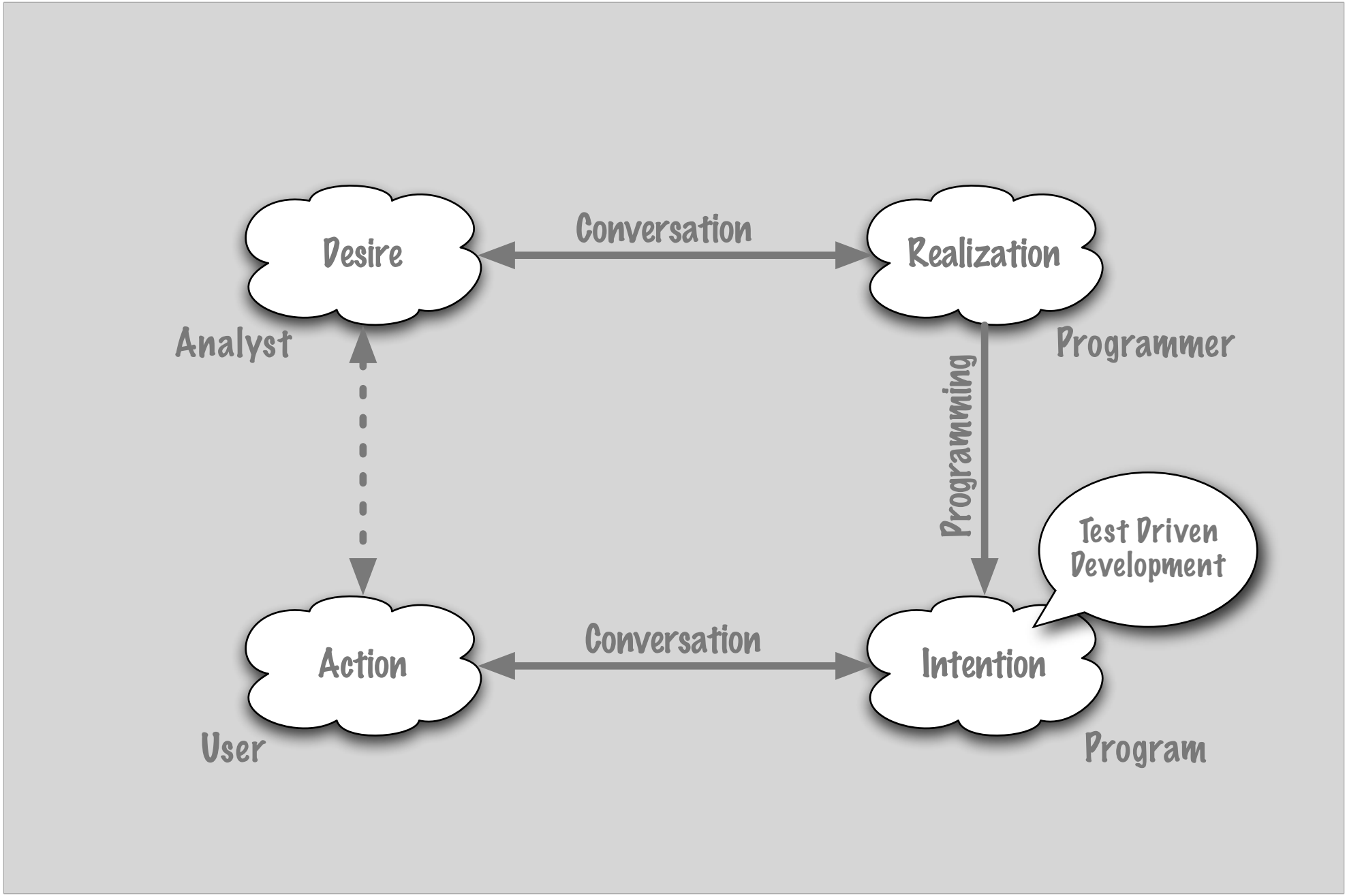
Program Visualization is about the program working.



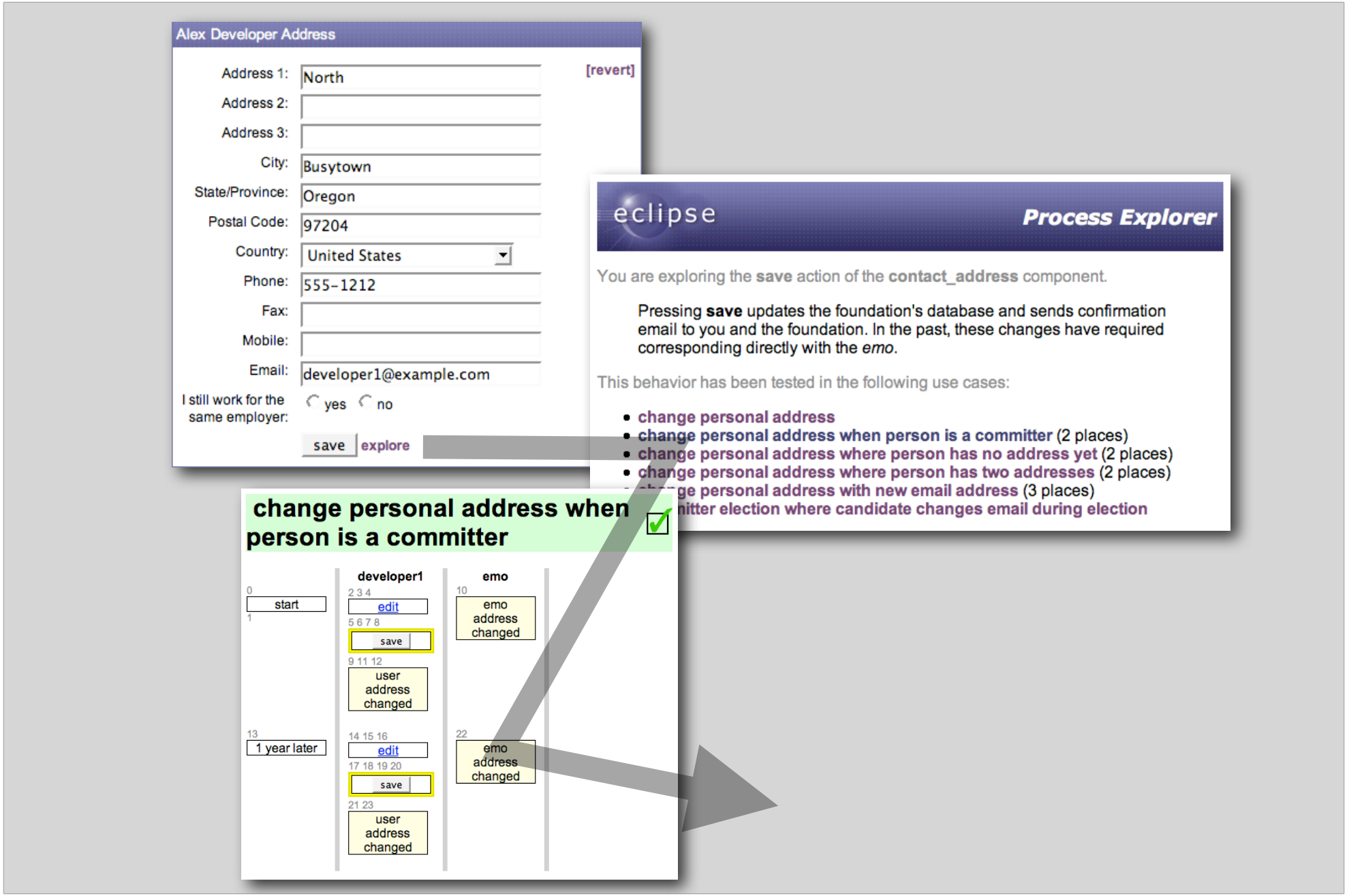
Social Visualization is about people working,
and their understanding of the work they do.



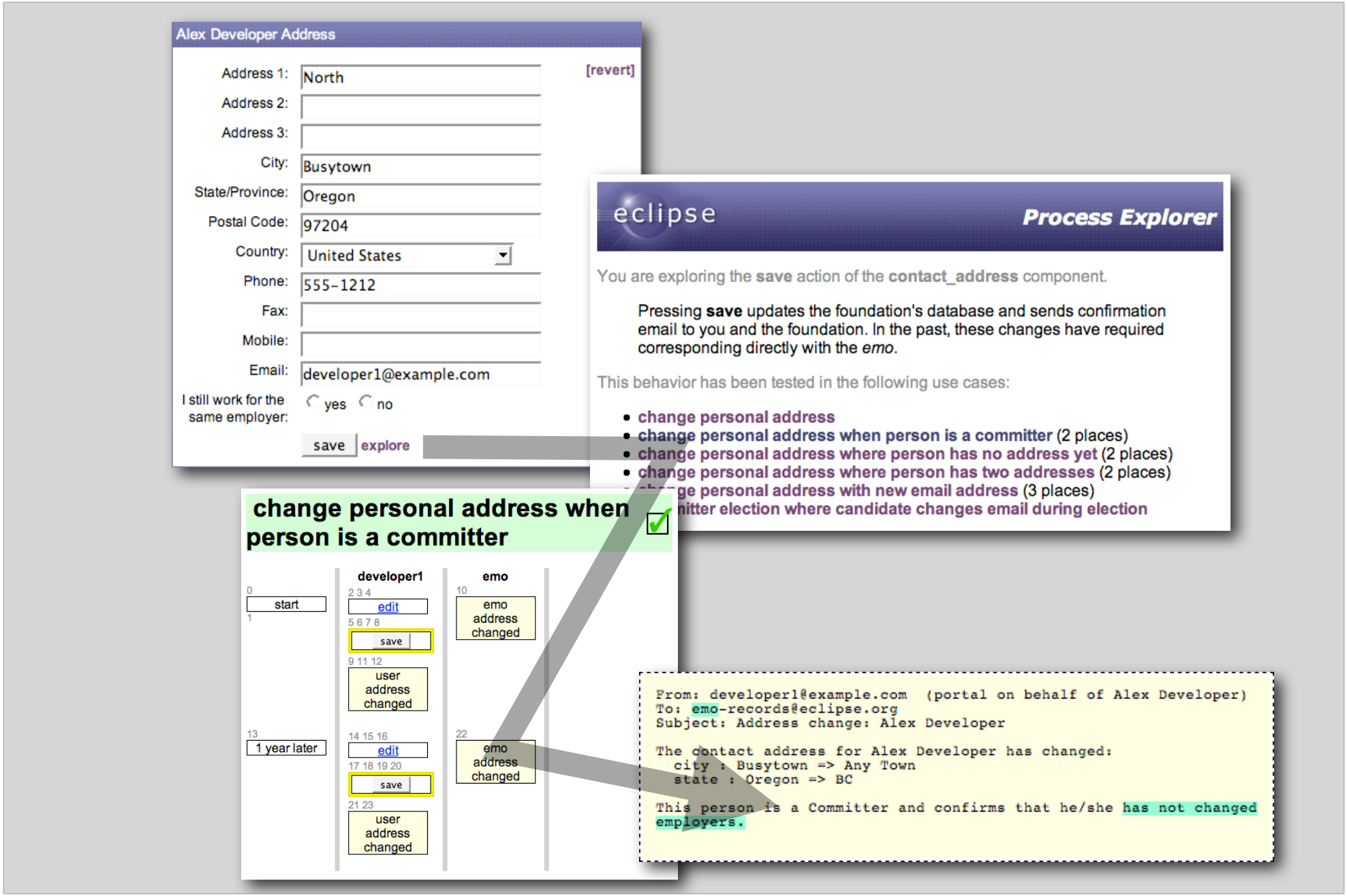
Question: What if filling out a form could be more like a **conversation** than an **interrogation**?



Question: What if filling out a form could be more like a **conversation** than an **interrogation**?



A user discovers how **employment status** is used when changing address.

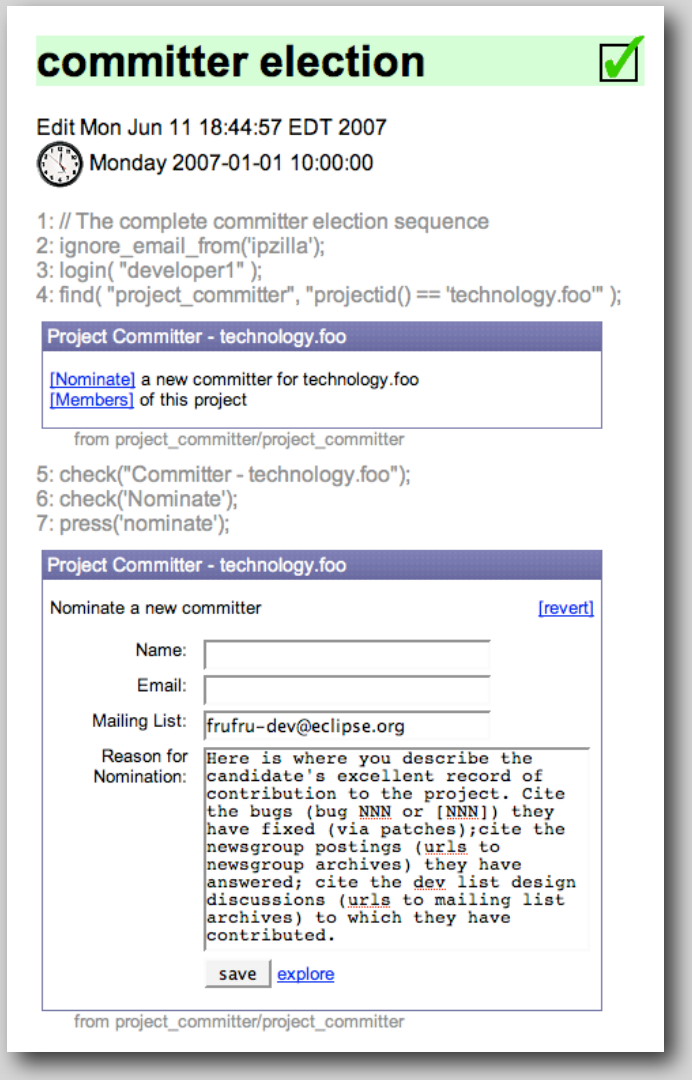


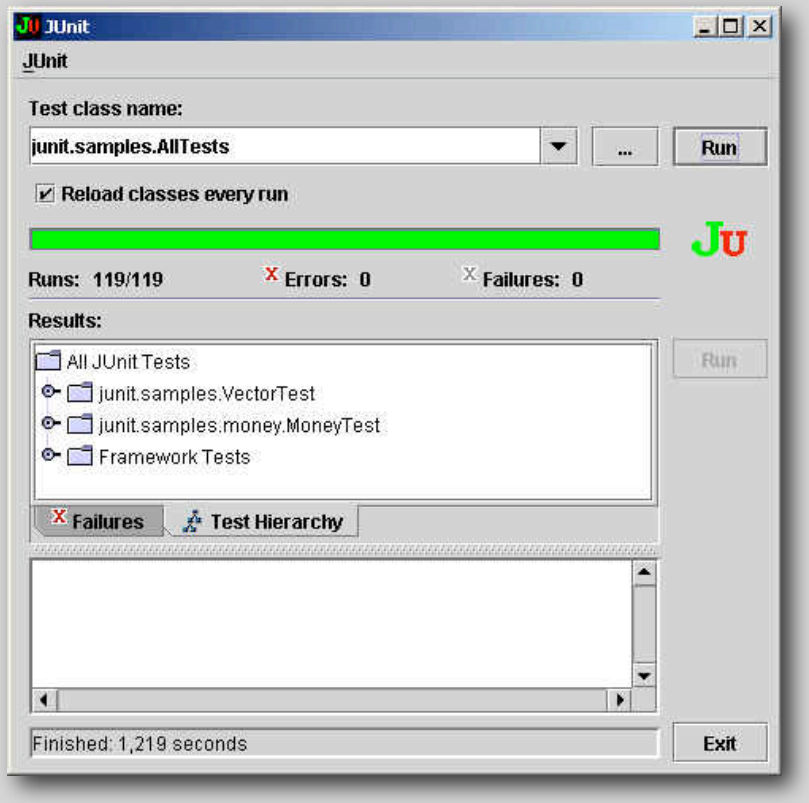
A user discovers how **employment status** is used when changing address.

The computer asks me a question.
I say, **why do you ask**?

I do not want the computer to pretend to answer me as a human. I will trust the computer that speaks computerish so long as I can make sense of what it says.

The computer's intention regarding your business can be learned by testing. I will save, index and format its test results so that you can learn what I learned while writing your programs.





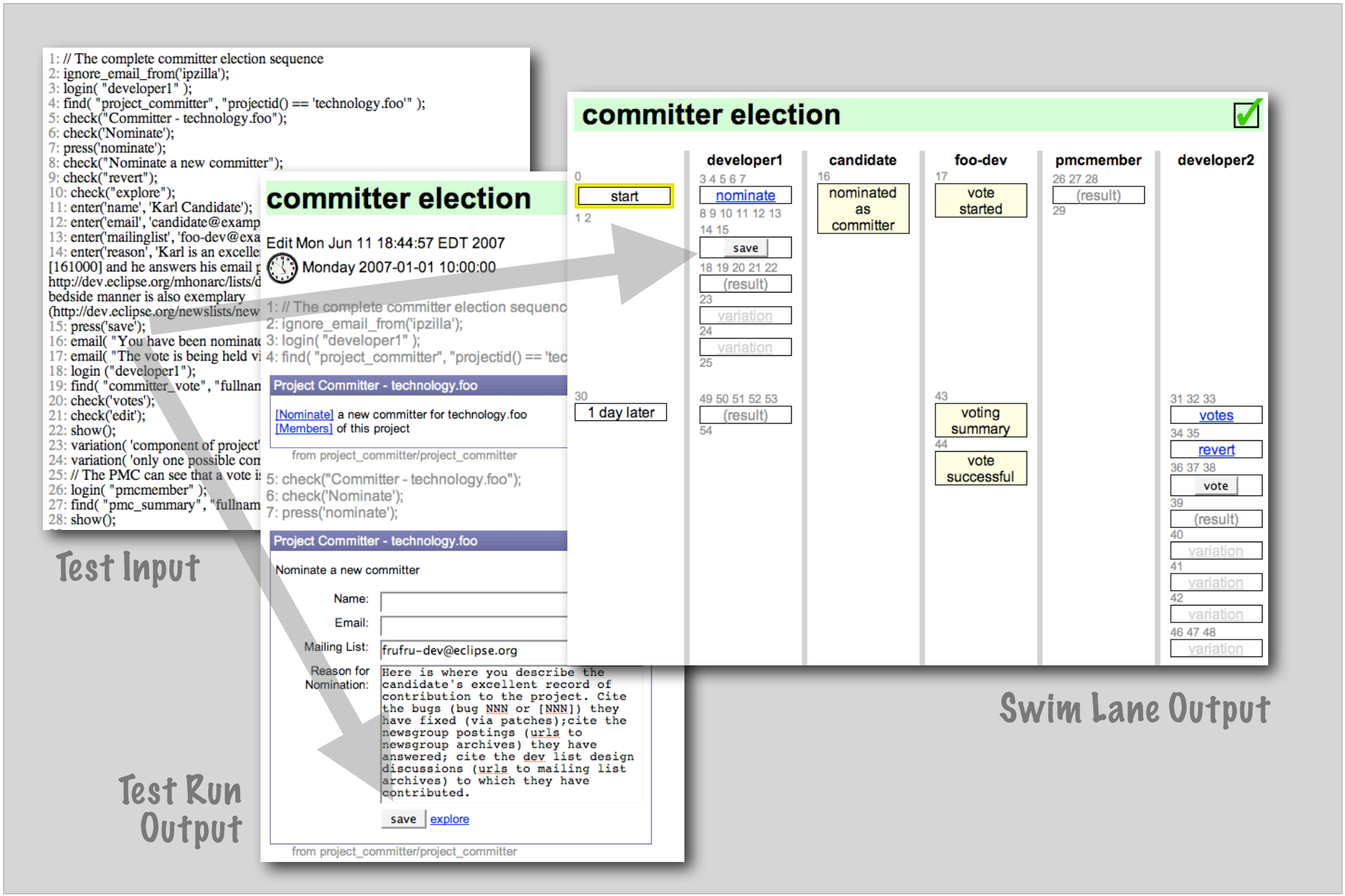
junit test runner

We choose to write tests at the **function** level and report output as a user would see it. We preserver junit's confidence inducing **green bar**.

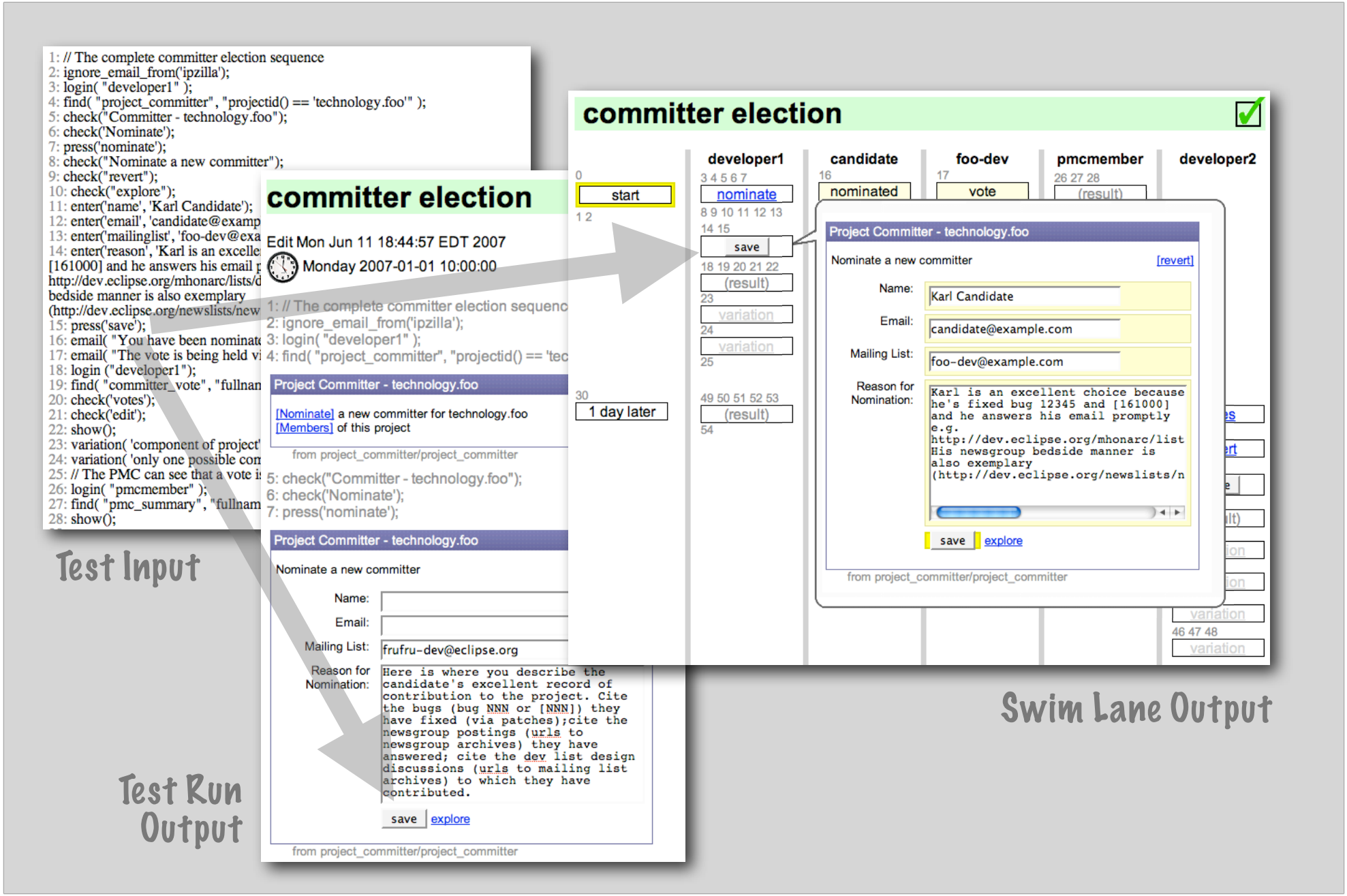
```
1: // The complete committer election sequence
2: ignore_email_from('ipzilla');
3: login( "developer1" );
4: find( "project_committer", "projectid() == 'technology.foo'" );
5: check("Committer - technology.foo");
6: check('Nominate');
7: press('nominate');
8: check("Nominate a new committer");
9: check("revert");
10: check("explore");
11: enter('name', 'Karl Candidate');
12: enter('email', 'candidate@example.com');
13: enter('mailinglist', 'foo-dev@example.com');
14: enter('reason', 'Karl is an excellent choice because he\'s fixed bug 12345 and
[161000] and he answers his email promptly e.g.
http://dev.eclipse.org/mhonarc/lists/dash-dev/msg00028.html. His newsgroup
bedside manner is also exemplary
(http://dev.eclipse.org/newslists/news.eclipse.technology.rap/msg341256.html).');
15: press('save');
16: email( "You have been nominated as a Committer", "candidate" );
17: email( "The vote is being held via the MyFoundation portal", "foo-dev" );
18: login ("developer1");
19: find( "committer_vote", "fullname() == 'Karl Candidate'" );
20: check('votes');
21: check('edit');
22: show();
23: variation( 'component of project' );
24: variation( 'only one possible component' );
25: // The PMC can see that a vote is going on
26: login( "pmcmember" );
27: find( "pmc_summary", "fullname() == 'Karl Candidate'" );
28: show();
```

Test Input

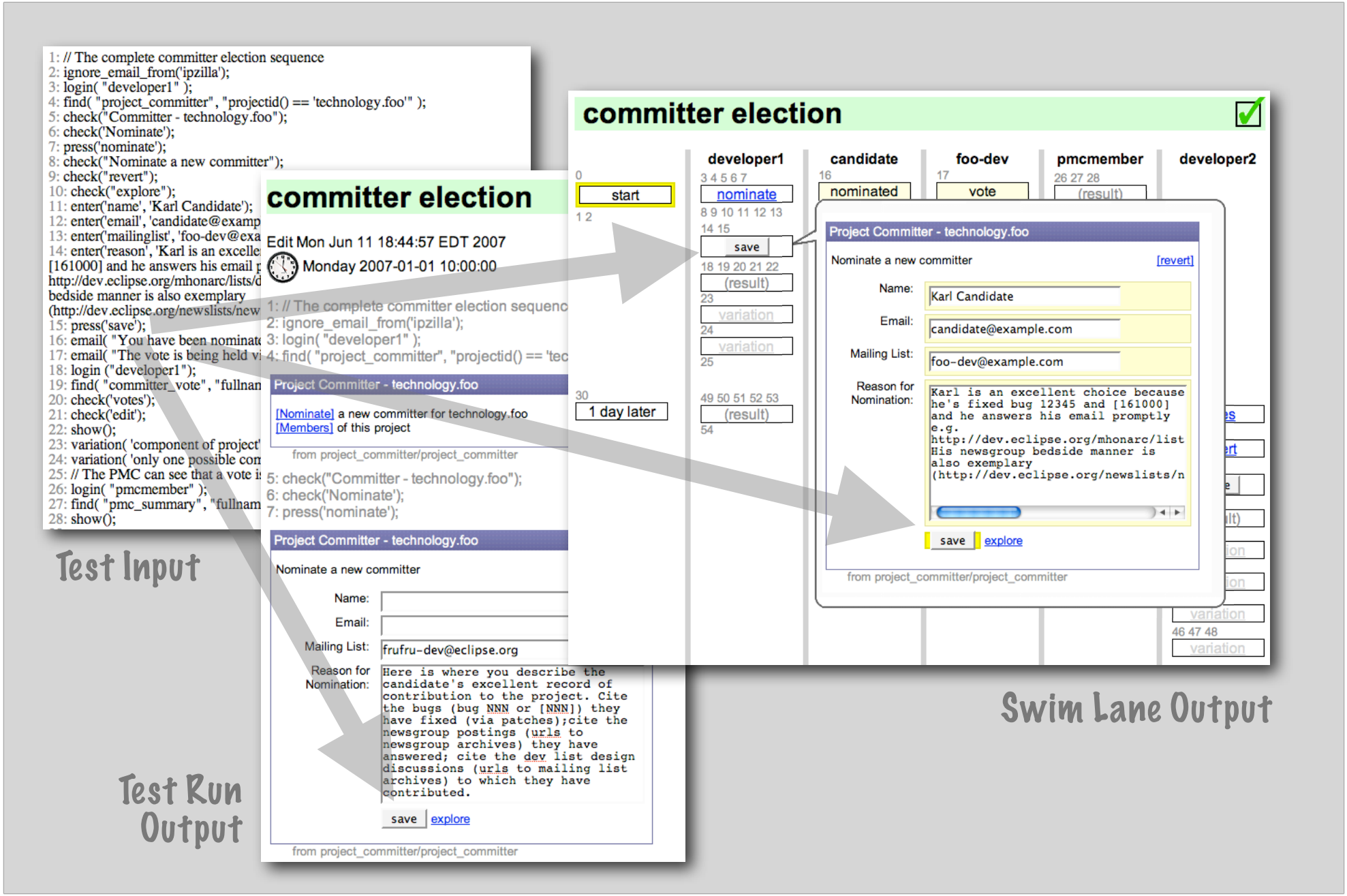
We write test cases in a domain specific language implemented as **functions** that operate the program.



A hypothetical operation, like pressing save, shows up as a **save button** in test output.



A hypothetical operation, like pressing save, shows up as a **save button** in test output.



A hypothetical operation, like pressing save, shows up as a **save button** in test output.

I will **demo** the Eclipse Foundation portal

We wrote php in the common style with only those accommodations to engineering practice as required by our testing methodology.

Do not think less of me for making the portal a simple program.

You will see that considerable activity occurs behind our plain screens. The foundation exposes all of this as part of its open and transparent policy.

The portal is judged **successful**

Six month's experience shows our portal easy to extend and deploy, often multiple times per day.

We have received neither complaints nor complements for using swim as online documentation.

Foundation staff trusts us to understand their work and would prefer we program all remaining processes with much haste.

The way forward with **social visualization**

We will present at the Pacific Northwest Software Quality Conference in Portland, Oregon, Fall 2007. We have been invited to present at the Agile Alliance Functional Testing Tools Visioning Workshop in the same timeframe.

Our implementation of swim remains specific to our portal. However, we expect to have the code released under the open source Eclipse Public License in time for the conference and would willingly help someone generalize our work.

The way forward with **conversational forms**

We believe some variation of swim suitable for tracing the complex data flows between the myriad of entities that make up our modern business world. We would like to see:

Extended SOA protocols to request and transfer swim related data for the portions of processes outside the realm of the originating computer.

An enlightened attitude toward the transparency implied by swim's comprehensive view of business process.