

The **swim** system for use case visualization

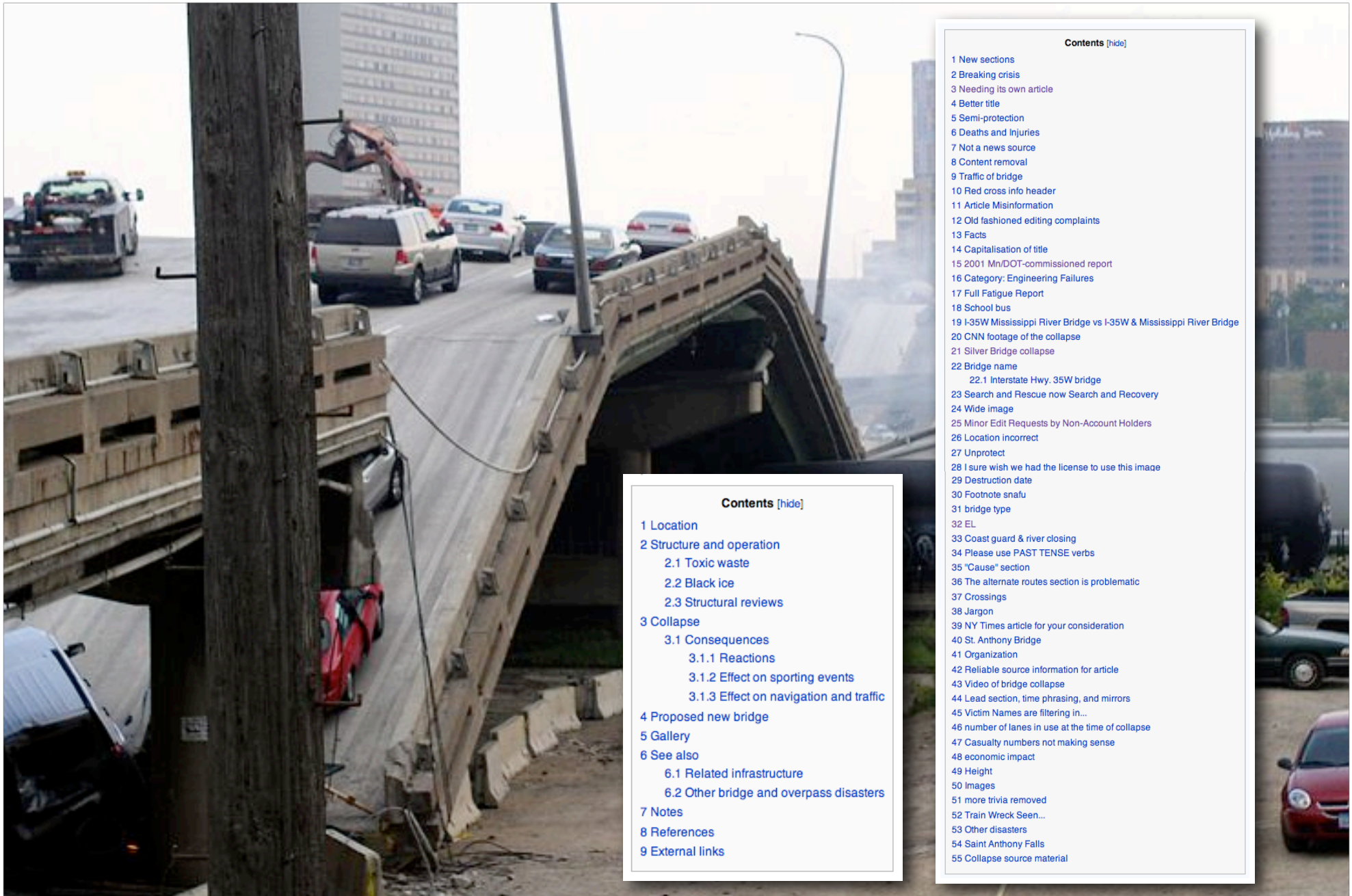
Ward Cunningham, AboutUs.org
Bjorn Freeman-Benson, Eclipse Foundation
Karl Matthias, Eclipse Foundaton

Released under (cc) creative commons, attribution, share-alike.

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.



Contents [\[hide\]](#)

- 1 Location
- 2 Structure and operation
 - 2.1 Toxic waste
 - 2.2 Black ice
 - 2.3 Structural reviews
- 3 Collapse
 - 3.1 Consequences
 - 3.1.1 Reactions
 - 3.1.2 Effect on sporting events
 - 3.1.3 Effect on navigation and traffic
- 4 Proposed new bridge
- 5 Gallery
- 6 See also
 - 6.1 Related infrastructure
 - 6.2 Other bridge and overpass disasters
- 7 Notes
- 8 References
- 9 External links

Contents [\[hide\]](#)

- 1 New sections
- 2 Breaking crisis
- 3 Needing its own article
- 4 Better title
- 5 Semi-protection
- 6 Deaths and Injuries
- 7 Not a news source
- 8 Content removal
- 9 Traffic of bridge
- 10 Red cross info header
- 11 Article Misinformation
- 12 Old fashioned editing complaints
- 13 Facts
- 14 Capitalisation of title
- 15 2001 Mn/DOT-commissioned report
- 16 Category: Engineering Failures
- 17 Full Fatigue Report
- 18 School bus
- 19 I-35W Mississippi River Bridge vs I-35W & Mississippi River Bridge
- 20 CNN footage of the collapse
- 21 Silver Bridge collapse
- 22 Bridge name
 - 22.1 Interstate Hwy. 35W bridge
- 23 Search and Rescue now Search and Recovery
- 24 Wide image
- 25 Minor Edit Requests by Non-Account Holders
- 26 Location incorrect
- 27 Unprotect
- 28 I sure wish we had the license to use this image
- 29 Destruction date
- 30 Footnote snafu
- 31 bridge type
- 32 EL
- 33 Coast guard & river closing
- 34 Please use PAST TENSE verbs
- 35 "Cause" section
- 36 The alternate routes section is problematic
- 37 Crossings
- 38 Jargon
- 39 NY Times article for your consideration
- 40 St. Anthony Bridge
- 41 Organization
- 42 Reliable source information for article
- 43 Video of bridge collapse
- 44 Lead section, time phrasing, and mirrors
- 45 Victim Names are filtering in...
- 46 number of lanes in use at the time of collapse
- 47 Casualty numbers not making sense
- 48 economic impact
- 49 Height
- 50 Images
- 51 more trivia removed
- 52 Train Wreck Seen...
- 53 Other disasters
- 54 Saint Anthony Falls
- 55 Collapse source material

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.

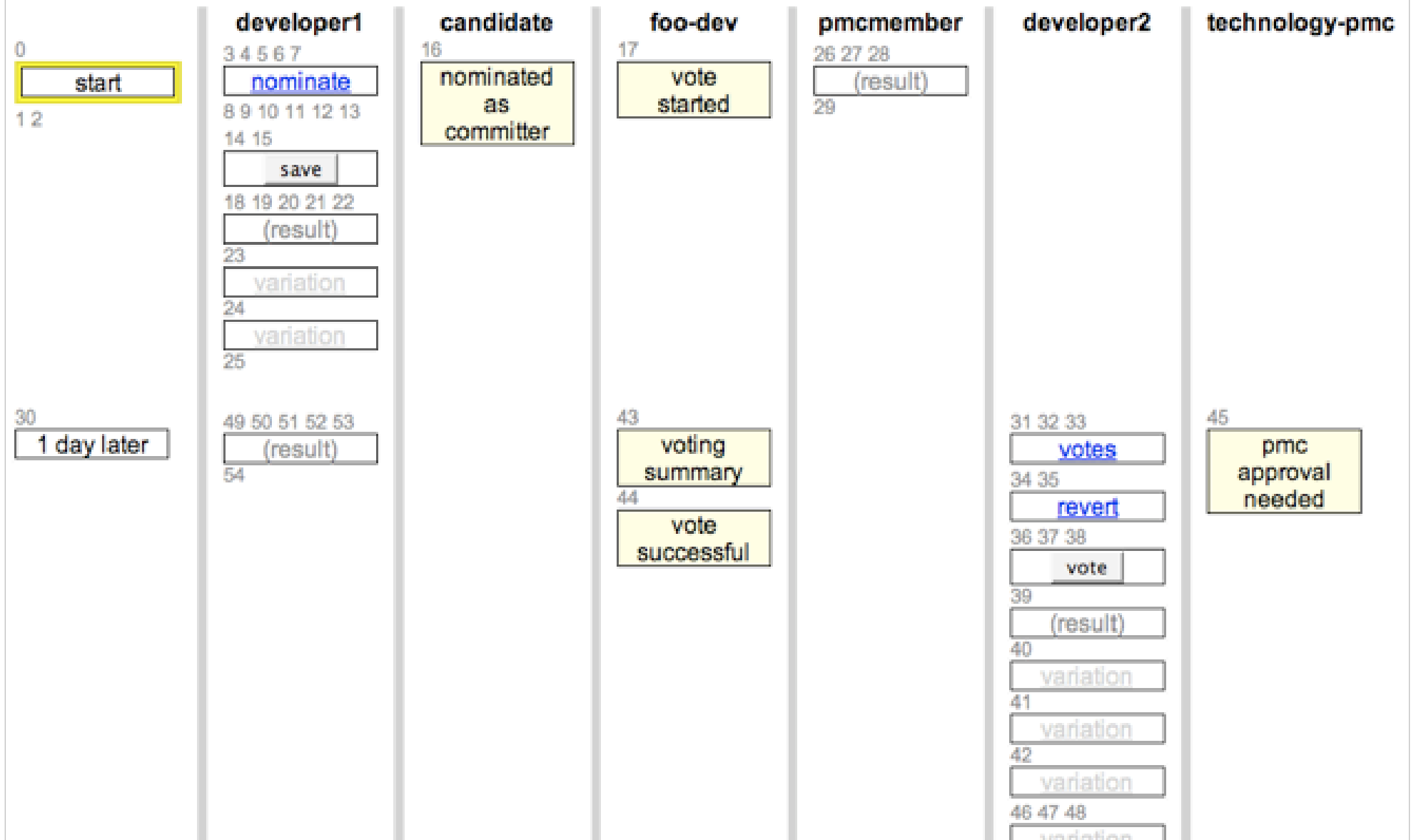


Bjorn and Karl reenact the decision making process that takes place in front of the **swim diagram** on a large computer screen.



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

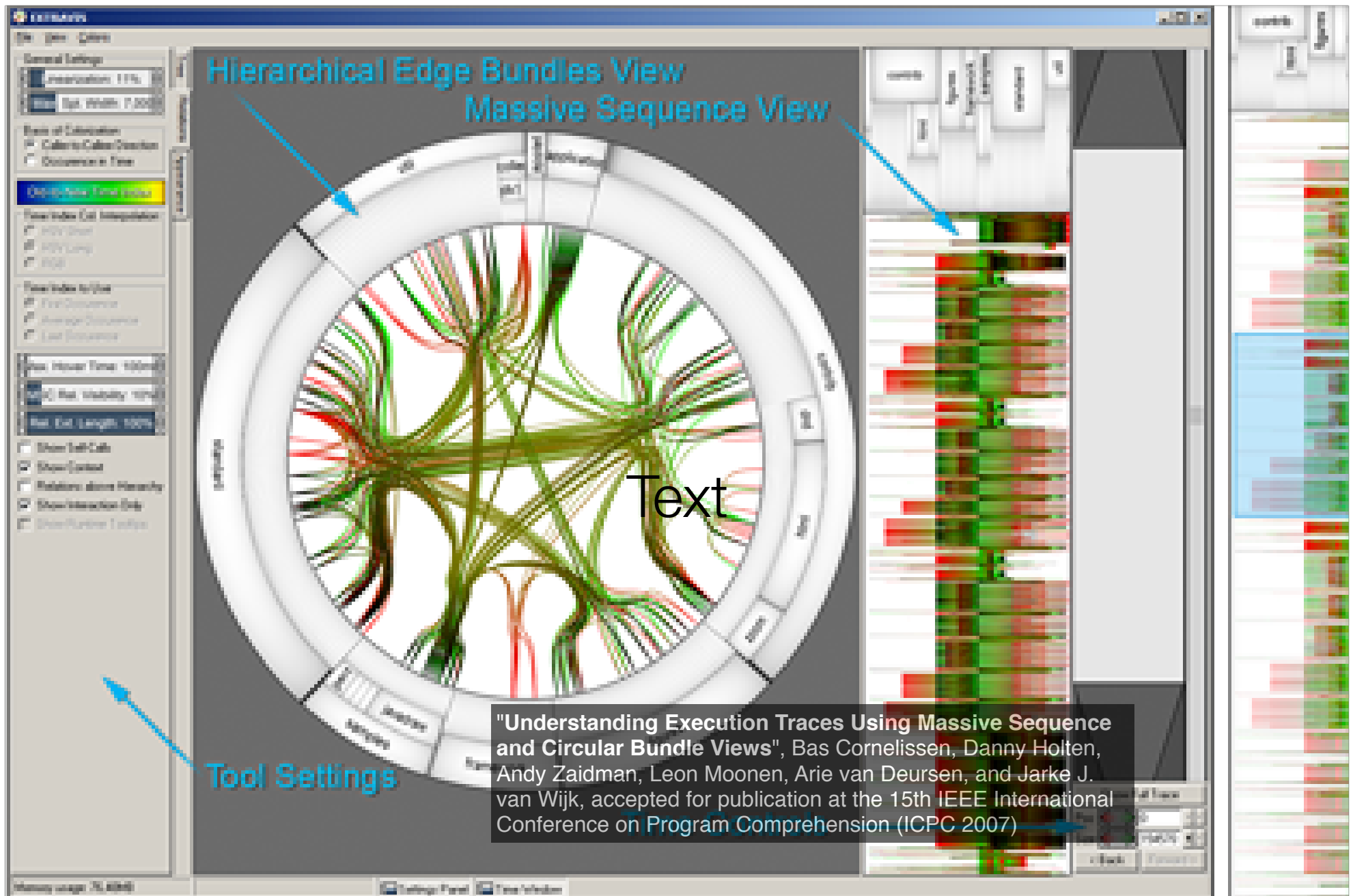
committer election



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



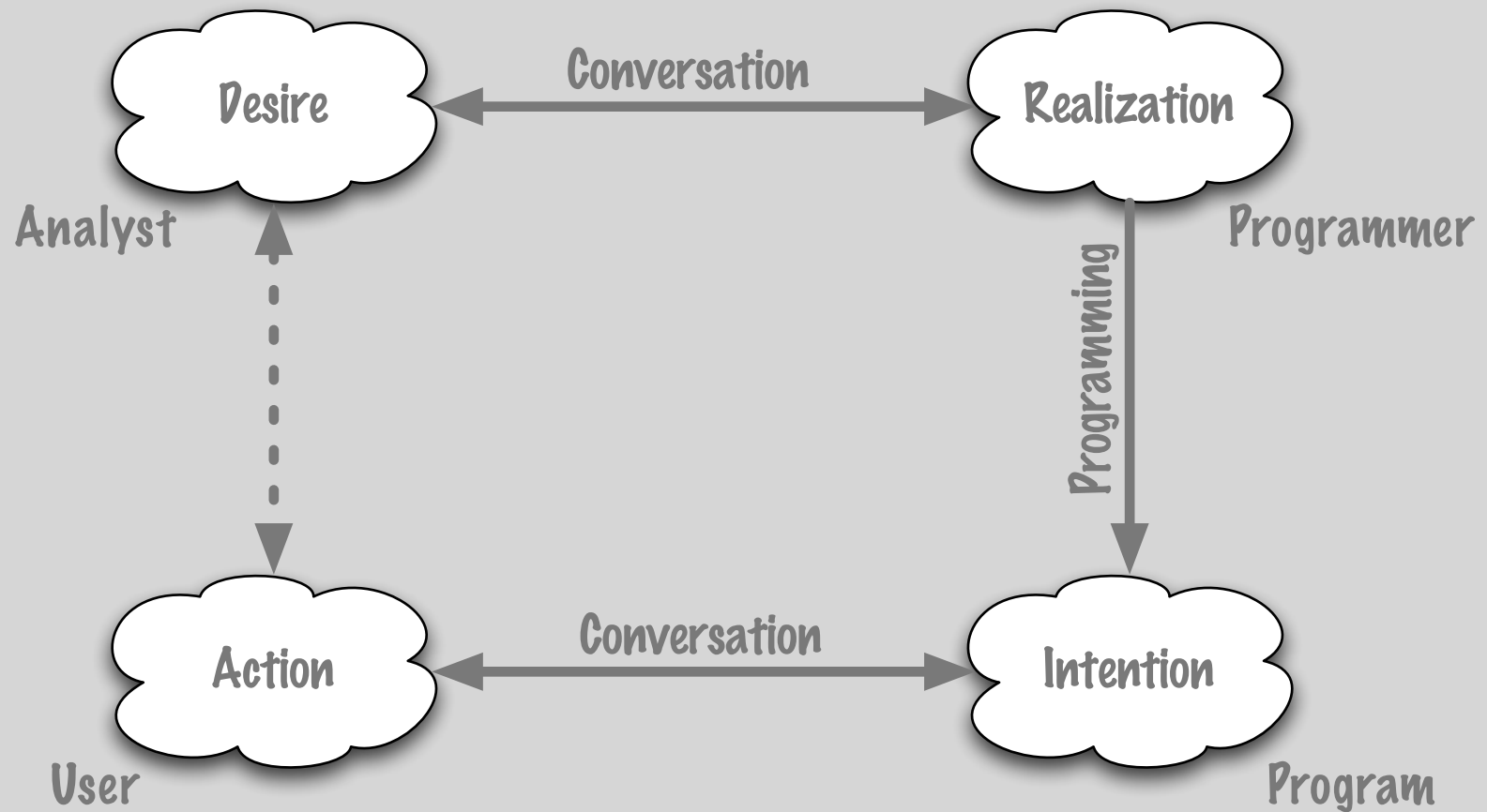
Business process engineers separate roles with bold lines suggesting swimming lanes.



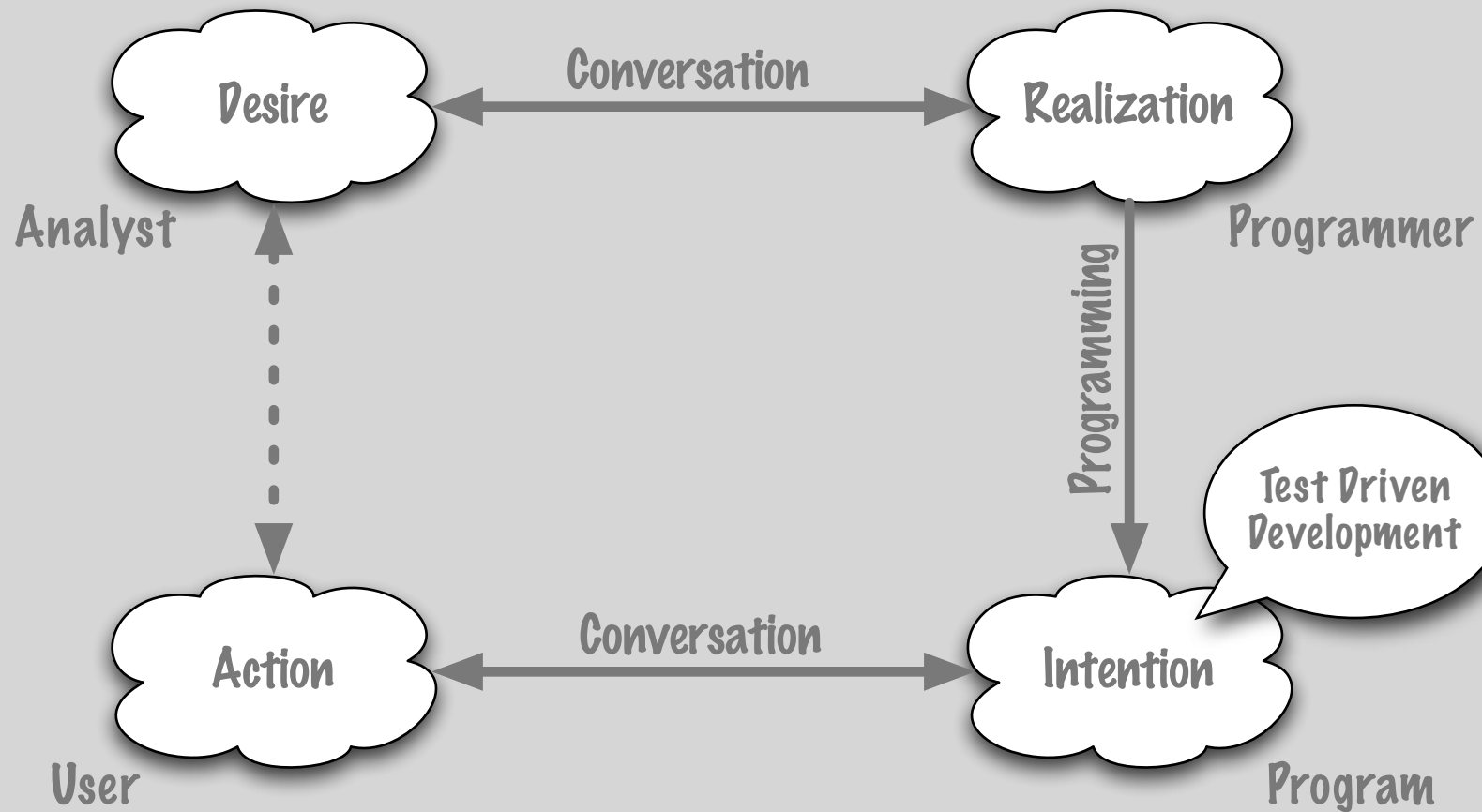
Program Visualization is about the program working.



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**?

Alex Developer Address [revert]

Address 1: North

Address 2:

Address 3:

City: Busytown

State/Province: Oregon

Postal Code: 97204

Country: United States

Phone: 555-1212

Fax:

Mobile:

Email: developer1@example.com

I still work for the same employer: yes no

eclipse **Process Explorer**

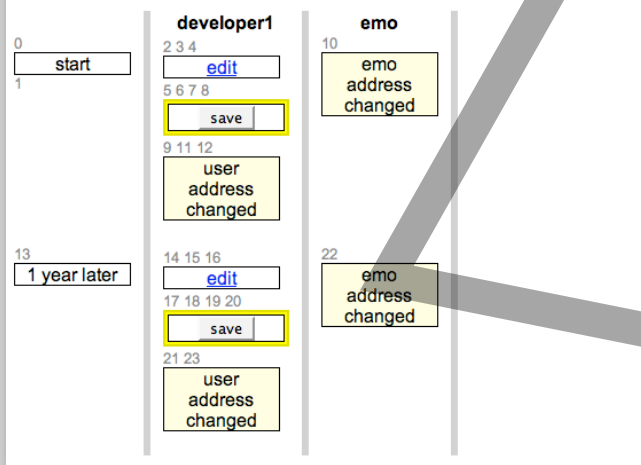
You are exploring the **save** action of the **contact_address** component.

Pressing **save** updates the foundation's database and sends confirmation email to you and the foundation. In the past, these changes have required corresponding directly with the *emo*.

This behavior has been tested in the following use cases:

- **change personal address**
- **change personal address when person is a committer** (2 places)
- **change personal address where person has no address yet** (2 places)
- **change personal address where person has two addresses** (2 places)
- **change personal address with new email address** (3 places)
- **committer election where candidate changes email during election**

change personal address when person is a committer



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

Alex Developer Address [revert]

Address 1:

Address 2:

Address 3:

City:

State/Province:

Postal Code:

Country:

Phone:

Fax:

Mobile:

Email:

I still work for the same employer: yes no

eclipse **Process Explorer**

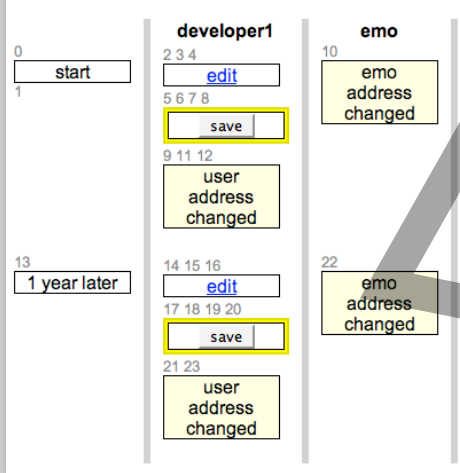
You are exploring the **save** action of the **contact_address** component.

Pressing **save** updates the foundation's database and sends confirmation email to you and the foundation. In the past, these changes have required corresponding directly with the *emo*.

This behavior has been tested in the following use cases:

- **change personal address**
- **change personal address when person is a committer** (2 places)
- **change personal address where person has no address yet** (2 places)
- **change personal address where person has two addresses** (2 places)
- **change personal address with new email address** (3 places)
- **committer election where candidate changes email during election**

change personal address when person is a committer ✓



```

From: developer1@example.com (portal on behalf of Alex Developer)
To: emo-records@eclipse.org
Subject: Address change: Alex Developer

The contact address for Alex Developer has changed:
city : Busytown => Any Town
state : Oregon => BC

This person is a Committer and confirms that he/she has not changed employers.

```

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.

committer election

Edit Mon Jun 11 18:44:57 EDT 2007



Monday 2007-01-01 10:00:00

```
1: // The complete committer election sequence
2: ignore_email_from('ipzilla');
3: login("developer1" );
4: find("project_committer", "projectId" == "technology.foo" );
```

Project Committer - technology.foo

[\[Nominate\]](#) a new committer for technology.foo
[\[Members\]](#) of this project

from project_committer/project_committer

```
5: check("Committer - technology.foo");
6: check('Nominate');
7: press('nominate');
```

Project Committer - technology.foo

Nominate a new committer [\[revert\]](#)

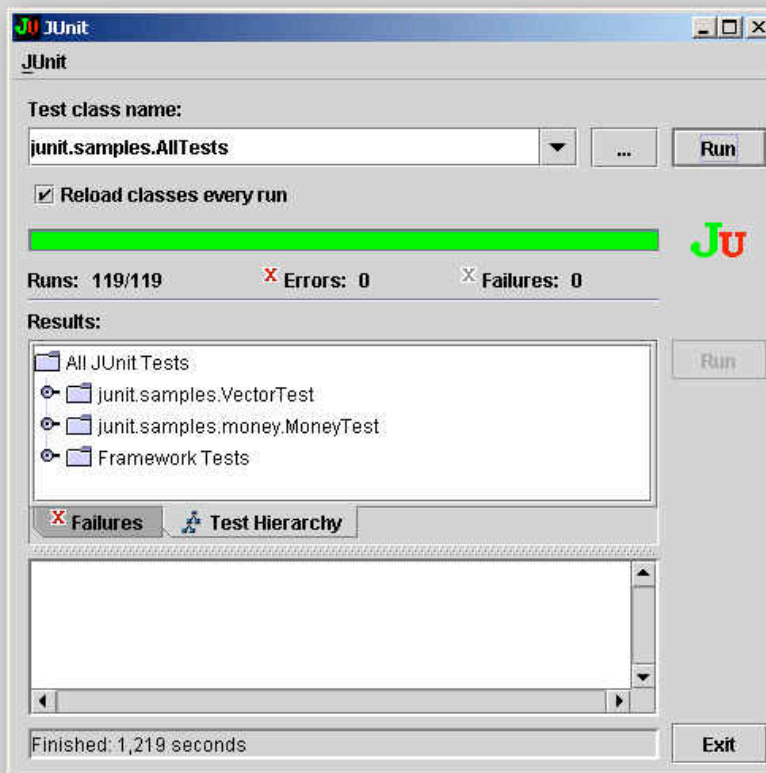
Name:

Email:

Mailing List:

Reason for Nomination:

from project_committer/project_committer



junit test runner

We choose to write tests at the **function** level and report output as a user would see it. We preserve 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.

```

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@exampl
13: enter('mailinglist', 'foo-dev@exa
14: enter('reason', 'Karl is an excell
[161000] and he answers his email p
http://dev.eclipse.org/mhonarc/lists/d
bedside manner is also exemplary
(http://dev.eclipse.org/newslists/new
15: press('save');
16: email("You have been nominat
17: email("The vote is being held vi
18: login("developer1");
19: find("committer_vote", "fullnam
20: check('votes');
21: check('edit');
22: show();
23: variation('component of project
24: variation('only one possible com
25: // The PMC can see that a vote is
26: login("pmcmember");
27: find("pmc_summary", "fullnam
28: show();

```

Test Input

Test Run Output

committer election

Edit Mon Jun 11 18:44:57 EDT 2007

Monday 2007-01-01 10:00:00

```

1: // The complete committer election sequenc
2: ignore_email_from('ipzilla');
3: login("developer1");
4: find("project_committer", "projectid() == 'tec
5: check("Committer - technology.foo");
6: check("Nominate");
7: press('nominate');

```

Project Committer - technology.foo

[\[Nominate\]](#) a new committer for technology.foo
[\[Members\]](#) of this project

from project_committer/project_committer

Nominate a new committer

Name:

Email:

Mailing List:

Reason for Nomination:

[explore](#)

from project_committer/project_committer

committer election

✓

<p>0 <input type="button" value="start"/></p> <p>12</p> <p>30 <input type="button" value="1 day later"/></p>	<p>developer1</p> <p>3 4 5 6 7 <input type="button" value="nominate"/></p> <p>8 9 10 11 12 13</p> <p>14 15 <input type="button" value="save"/></p> <p>18 19 20 21 22 <input type="button" value="(result)"/></p> <p>23 <input type="button" value="variation"/></p> <p>24 <input type="button" value="variation"/></p> <p>25</p> <p>49 50 51 52 53 <input type="button" value="(result)"/></p> <p>54</p>	<p>candidate</p> <p>16 nominated as committer</p>	<p>foo-dev</p> <p>17 <input type="button" value="vote started"/></p> <p>43 <input type="button" value="voting summary"/></p> <p>44 <input type="button" value="vote successful"/></p>	<p>pmcmember</p> <p>26 27 28 <input type="button" value="(result)"/></p> <p>29</p>	<p>developer2</p> <p>31 32 33 <input type="button" value="votes"/></p> <p>34 35 <input type="button" value="revert"/></p> <p>36 37 38 <input type="button" value="vote"/></p> <p>39 <input type="button" value="(result)"/></p> <p>40 <input type="button" value="variation"/></p> <p>41 <input type="button" value="variation"/></p> <p>42 <input type="button" value="variation"/></p> <p>46 47 48 <input type="button" value="variation"/></p>
--	---	--	--	---	--

Swim Lane Output

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

```

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@exam
13: enter('mailinglist', 'foo-dev@exa
14: enter('reason', 'Karl is an excel
[161000] and he answers his email p
http://dev.eclipse.org/mhonarc/lists/d
bedside manner is also exemplary
(http://dev.eclipse.org/newslists/new
15: press('save');
16: email("You have been nominat
17: email("The vote is being held vi
18: login("developer1");
19: find("committer_vote", "fullnam
20: check('votes');
21: check('edit');
22: show();
23: variation('component of project
24: variation('only one possible com
25: // The PMC can see that a vote is
26: login("pmcmember");
27: find("pmc_summary", "fullnam
28: show();

```

Test Input

Test Run Output

committer election

Edit Mon Jun 11 18:44:57 EDT 2007

Monday 2007-01-01 10:00:00

```

1: // The complete committer election sequenc
2: ignore_email_from('ipzilla');
3: login("developer1");
4: find("project_committer", "projectid() == 'tec
5: check("Committer - technology.foo");
6: check("Nominate");
7: press('nominate');

```

Project Committer - technology.foo

[\[Nominate\]](#) a new committer for technology.foo
[\[Members\]](#) of this project

from project_committer/project_committer

Project Committer - technology.foo

Nominate a new committer

Name:

Email:

Mailing List: frufu-dev@eclipse.org

Reason for Nomination: Here is where you describe the candidate's excellent record of contribution to the project. Cite the bugs (bug NNN or [NNN]) they have fixed (via patches);cite the newsgroup postings (urls to newsgroup archives) they have answered; cite the dev list design discussions (urls to mailing list archives) to which they have contributed.

from project_committer/project_committer

committer election

✓

0	16	17	26 27 28
start	nominated	vote	(result)
12	3 4 5 6 7 nominate	8 9 10 11 12 13 <input type="button" value="save"/>	14 15 <input type="button" value="(result)"/>
30	18 19 20 21 22 <input type="button" value="variation"/>	23 <input type="button" value="variation"/>	24 <input type="button" value="variation"/>
1 day later	25 <input type="button" value="(result)"/>	49 50 51 52 53 <input type="button" value="(result)"/>	54 <input type="button" value="(result)"/>

Project Committer - technology.foo

Nominate a new committer [\[revert\]](#)

Name:

Email:

Mailing List:

Reason for Nomination:

from project_committer/project_committer

Swim Lane Output

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

```

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@exampl
13: enter('mailinglist', 'foo-dev@exa
14: enter('reason', 'Karl is an excell
[161000] and he answers his email p
http://dev.eclipse.org/mhonarc/lists/d
bedside manner is also exemplary
(http://dev.eclipse.org/newslists/new
15: press('save');
16: email("You have been nominat
17: email("The vote is being held vi
18: login("developer1");
19: find("committer_vote", "fullnam
20: check('votes');
21: check('edit');
22: show();
23: variation('component of project
24: variation('only one possible com
25: // The PMC can see that a vote is
26: login("pmcmember");
27: find("pmc_summary", "fullnam
28: show();

```

Test Input

Test Run Output

committer election

Edit Mon Jun 11 18:44:57 EDT 2007

Monday 2007-01-01 10:00:00

```

1: // The complete committer election sequenc
2: ignore_email_from('ipzilla');
3: login("developer1");
4: find("project_committer", "projectid() == 'tec
5: check("Committer - technology.foo");
6: check("Nominate");
7: press('nominate');

```

Project Committer - technology.foo

[\[Nominate\]](#) a new committer for technology.foo
[\[Members\]](#) of this project

from project_committer/project_committer

Project Committer - technology.foo

Nominate a new committer

Name:

Email:

Mailing List:

Reason for Nomination:

from project_committer/project_committer

committer election

0	developer1	candidate	foo-dev	pmcmember	developer2
3 4 5 6 7	<input type="button" value="nominate"/>	<input type="button" value="nominated"/>	<input type="button" value="vote"/>	<input type="button" value="(result)"/>	
8 9 10 11 12 13	<input type="button" value="save"/>				
14 15	<input type="button" value="(result)"/>				
16	<input type="button" value="variation"/>				
17	<input type="button" value="variation"/>				
18 19 20 21 22	<input type="button" value="(result)"/>				
23					
24					
25					
26 27 28				<input type="button" value="(result)"/>	
29					
30	<input type="button" value="1 day later"/>				
31					
32					
33					
34					
35					
36					
37					
38					
39					
40					
41					
42					
43					
44					
45					
46 47 48					
49					
50					
51					
52					
53					
54					
55					

Project Committer - technology.foo

Nominate a new committer [\[revert\]](#)

Name:

Email:

Mailing List:

Reason for Nomination:

from project_committer/project_committer

Swim Lane 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.