

SATTtoSE 2009

End-user Web Development Lively Fabrik and Lively Wiki

Jens Lincke

Robert Krahn

Robert Hirschfeld

Hasso-Plattner-Institut Potsdam

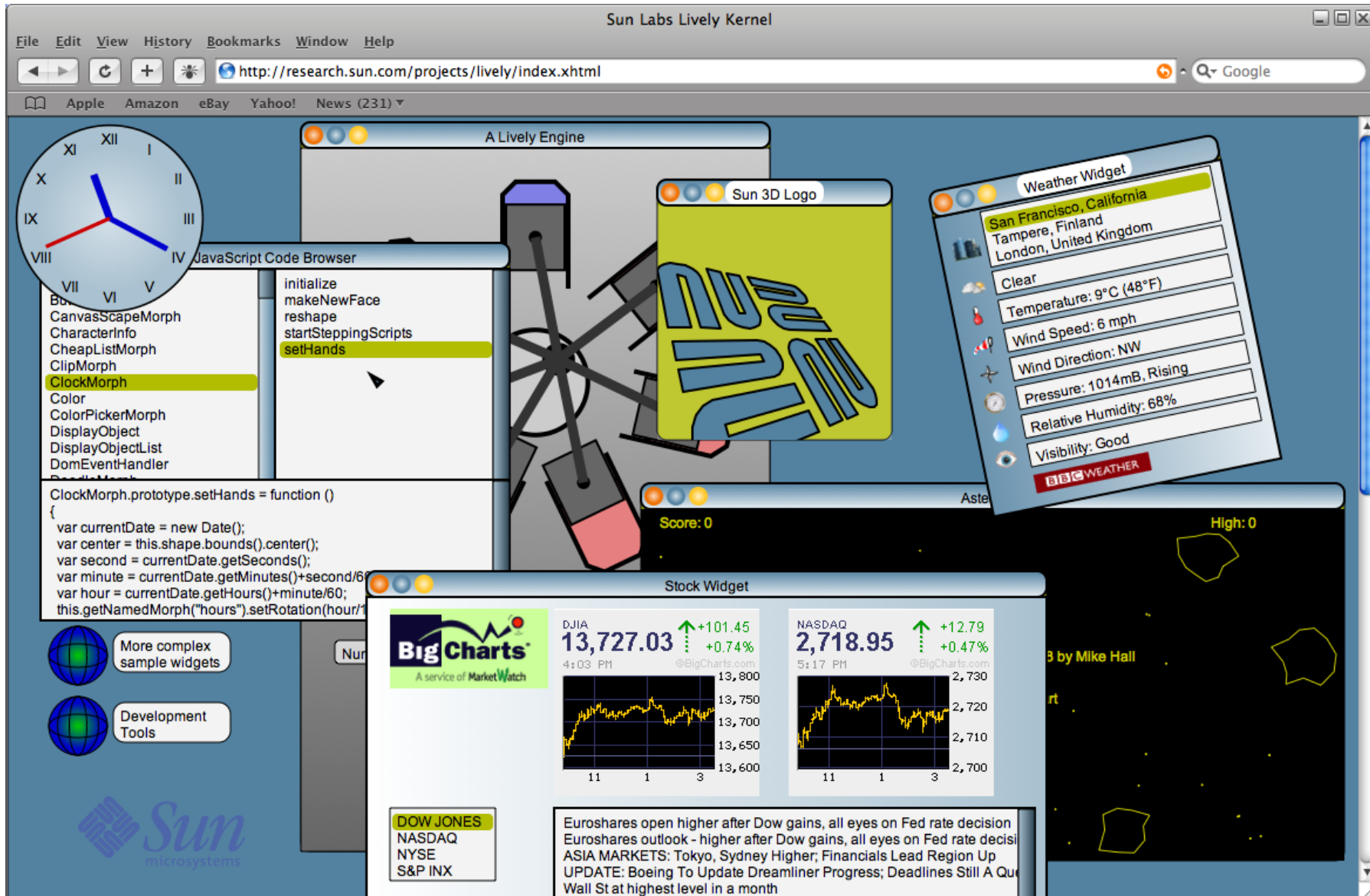
Software Architecture Group

<http://www.hpi.uni-potsdam.de/swa/>

Motivation

- End-users treat Web browser as OS
- EUD scenarios
 - Customization of existing Web applications
 - Widgets
 - Mashups
 - Simple Web Applications
- Growing Web programming capacities

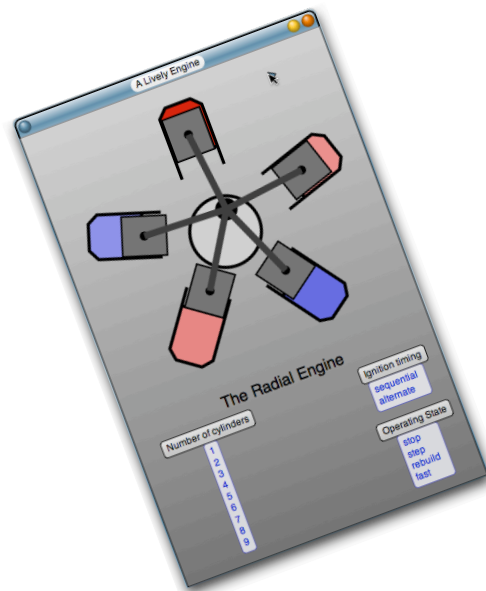
Background: Lively Kernel



Lively Kernel Vision

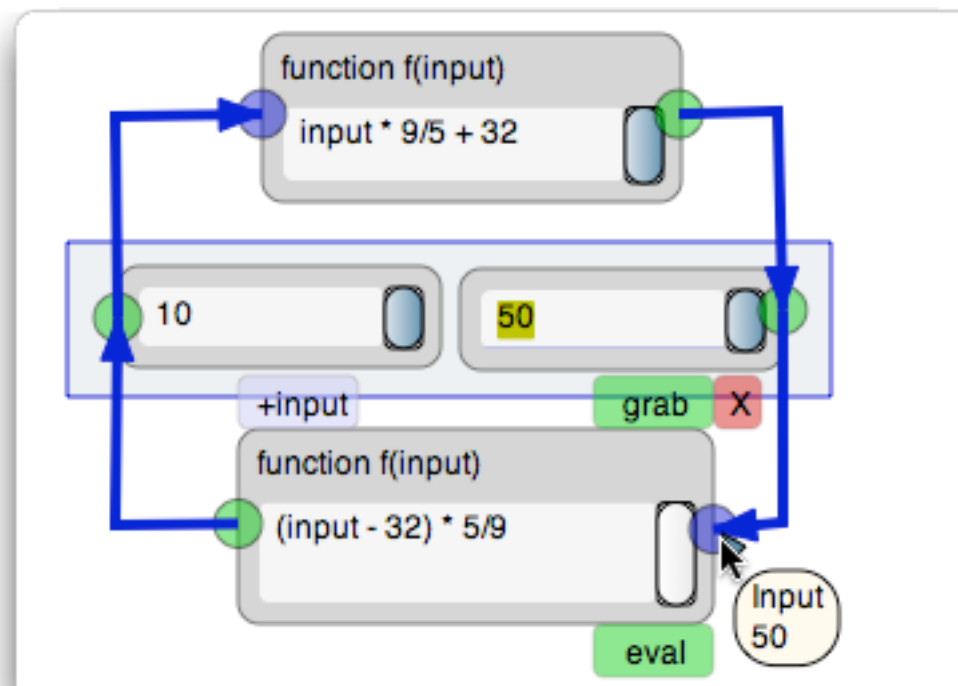
A wiki of Active Objects that can be programmed in JavaScript by wires and tiles

[Dan Ingalls]



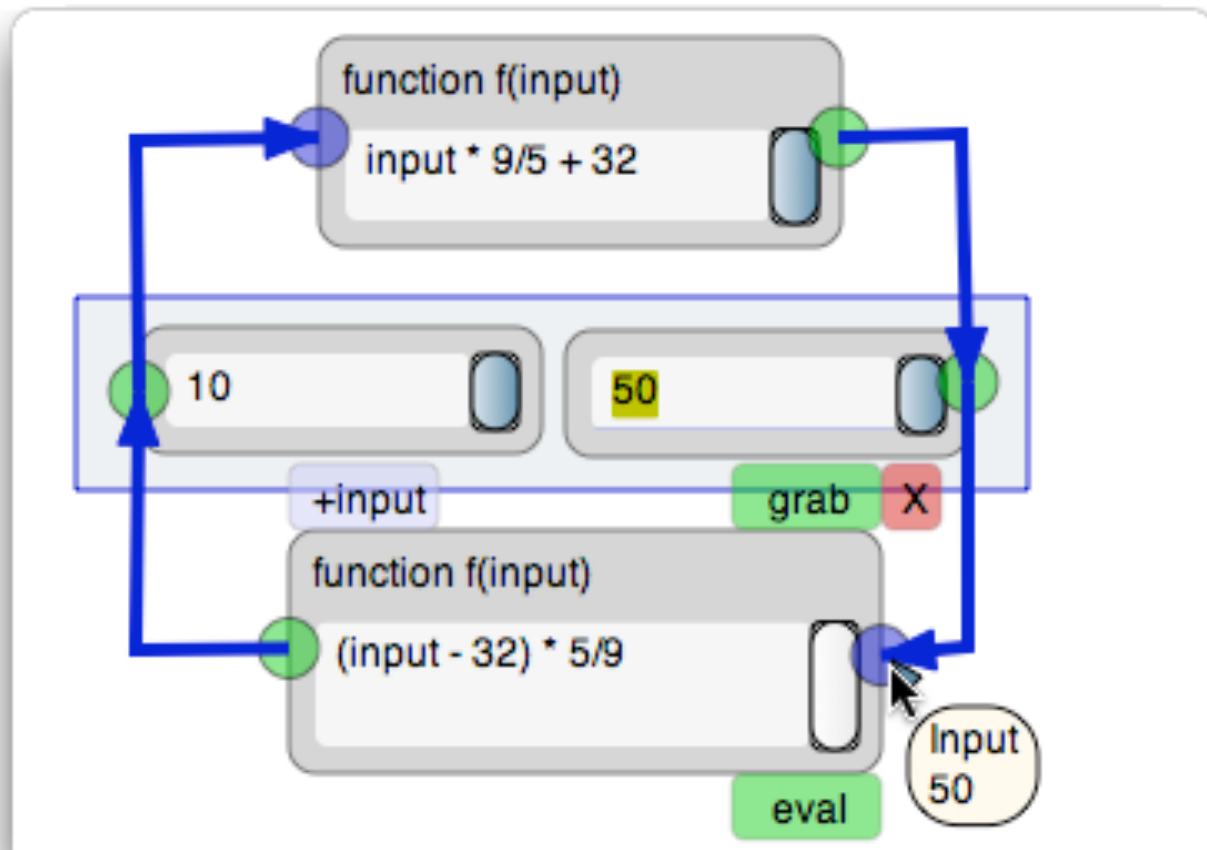
Lively Fabrik

- Empower end-users to create dynamic Web-applications with Mashups and Active Objects
- Combine Fabrik and Lively Kernel

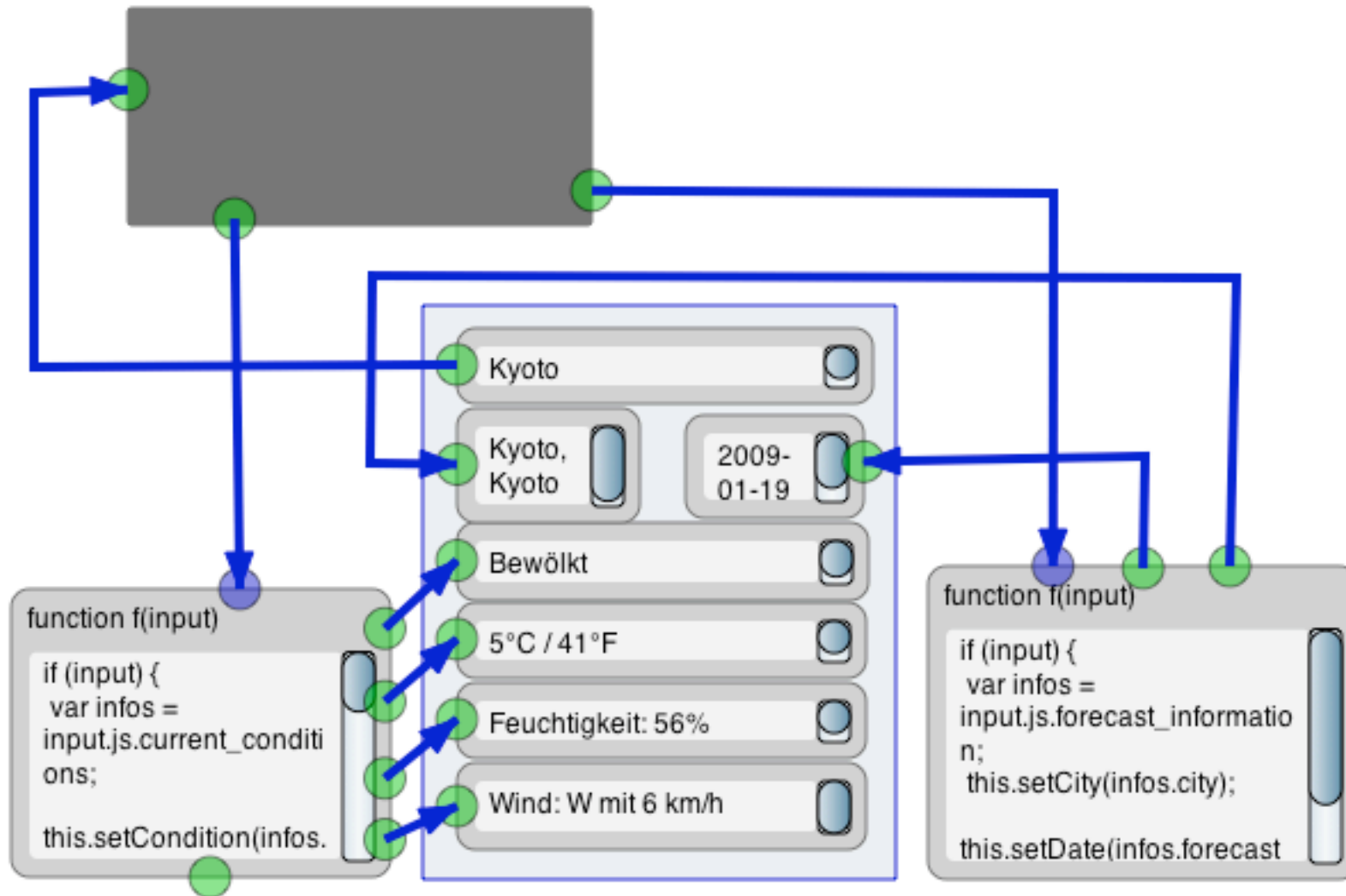


User Interface

- Combine UI and Behavior
- Continuous Running
- Visual Aids



Example Weather Widget



Lively Wiki A Development Environment for Creating and Sharing Rich Web Content

- Robert Krahn's Master's thesis
- Wiki principles: open, incremental, organic
- Developing *inside the Web for the Web*

Welcome to the Lively Kernel Wiki


Introduction
 The Sun Labs Lively Kernel is a web programming environment developed at Sun Microsystems Laboratories. The Lively Kernel supports desktop-style applications with rich graphics and direct manipulation capabilities, but without the installation or upgrade hassles that conventional desktop applications have. The system is written entirely in the JavaScript programming language, a language supported by all the web browsers, with the intent that the system can run in commercial web browsers without installation or any plug-in components. The system leverages the dynamic characteristics of the JavaScript language to make it possible to create, modify and deploy applications on the fly, using tools built into the system itself. In addition to its application execution capabilities, the Lively Kernel can also function as an integrated development environment (IDE), making the whole system self-sufficient and able to improve and extend itself dynamically.

Useful Links:
[Dan's Kaleidoscope](#)
[Tutorial: Programming in the wiki](#)
[Lively Examples](#)
[Wiki control scripts](#)
[Fabrik playground](#)
[Engine example](#)
[Simple objects](#)
[Project Page at Sun](#)

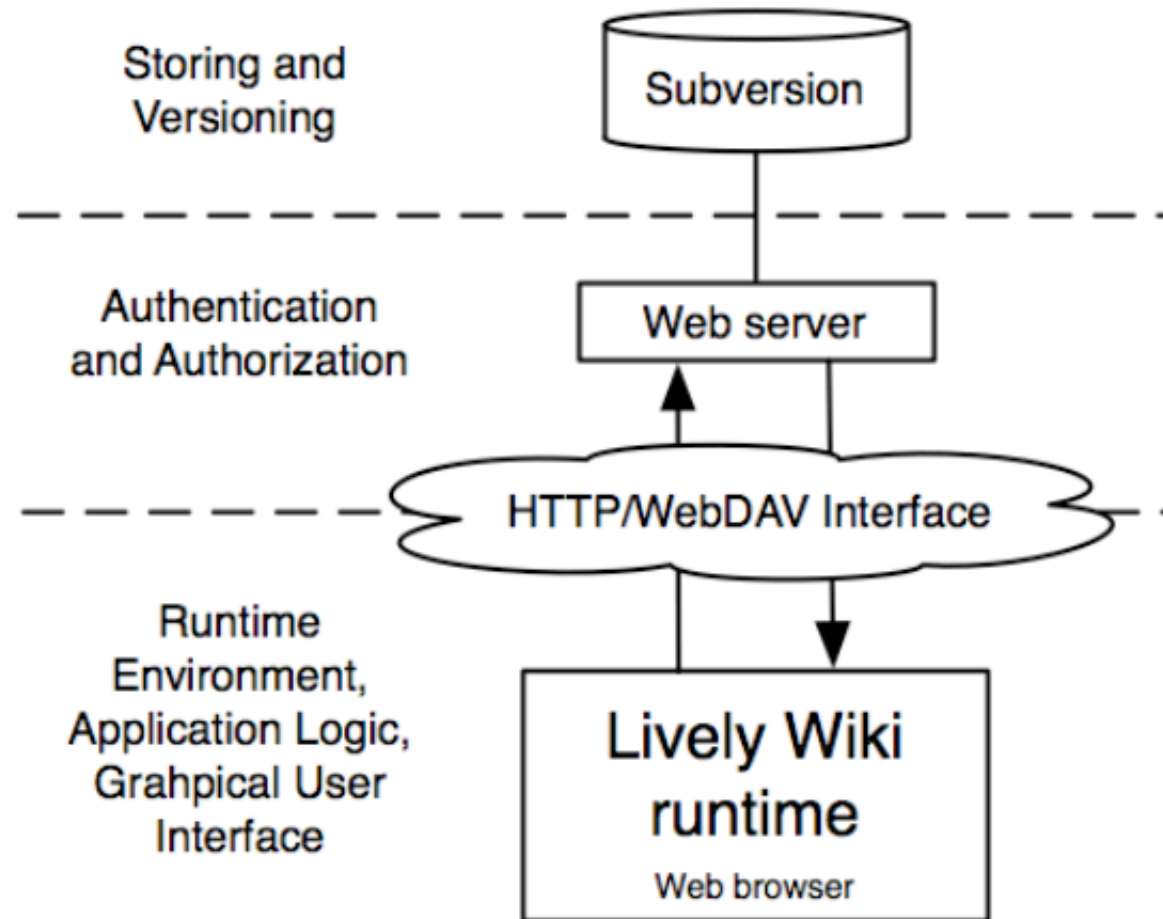
Latest changes

Refresh Filename filter

```
rk-p01test.html (15800 -- robertkrahn)
tips2.html (15468 -- philmaker)
pweaver.html (15467 -- philmaker)
pweaver-test3.html (15465 -- philmaker)
pweaver-test2.html (15463 -- philmaker)
pweaver-widgets.html (15437 -- philmaker)
twx.html (15435 -- Dan)
swank.html (15434 -- philmaker)
rk-calApril.html (15430 -- robertkrahn)
rk-index.html (15428 -- robertkrahn)
rk-p06.html (15426 -- robertkrahn)
rk-p04.html (15424 -- robertkrahn)
```

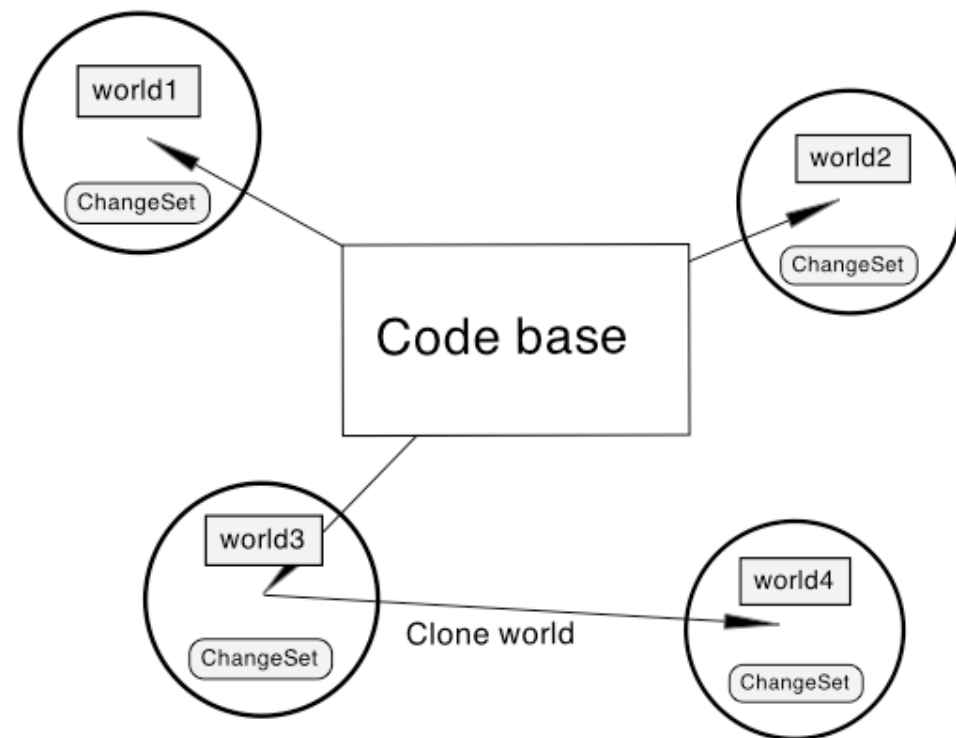


Lively Wiki Architecture

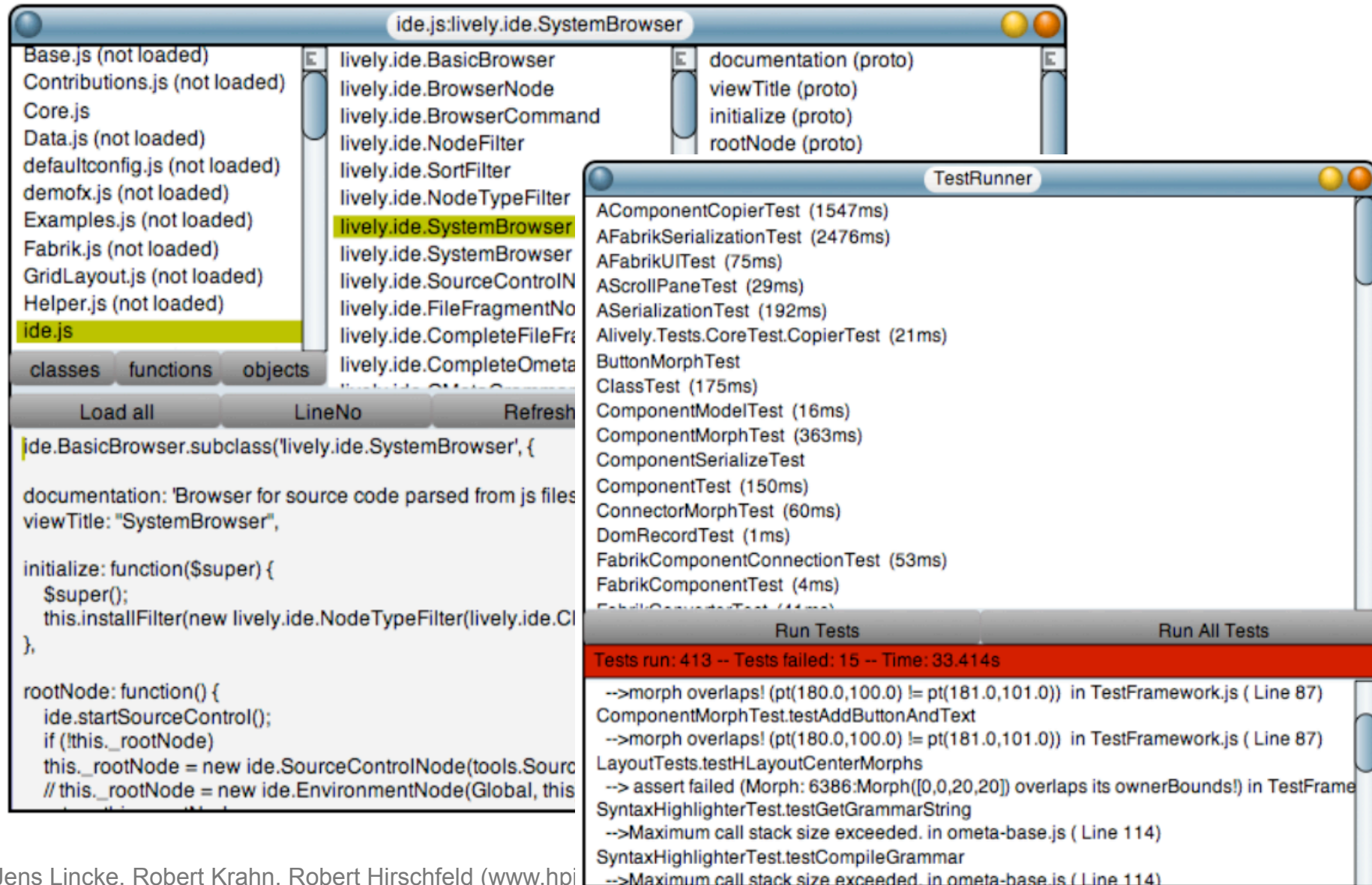


Web Programming Model

- Network of Lively Kernel worlds
 - Versions
 - Links
- Development tools
 - JavaScript and beyond



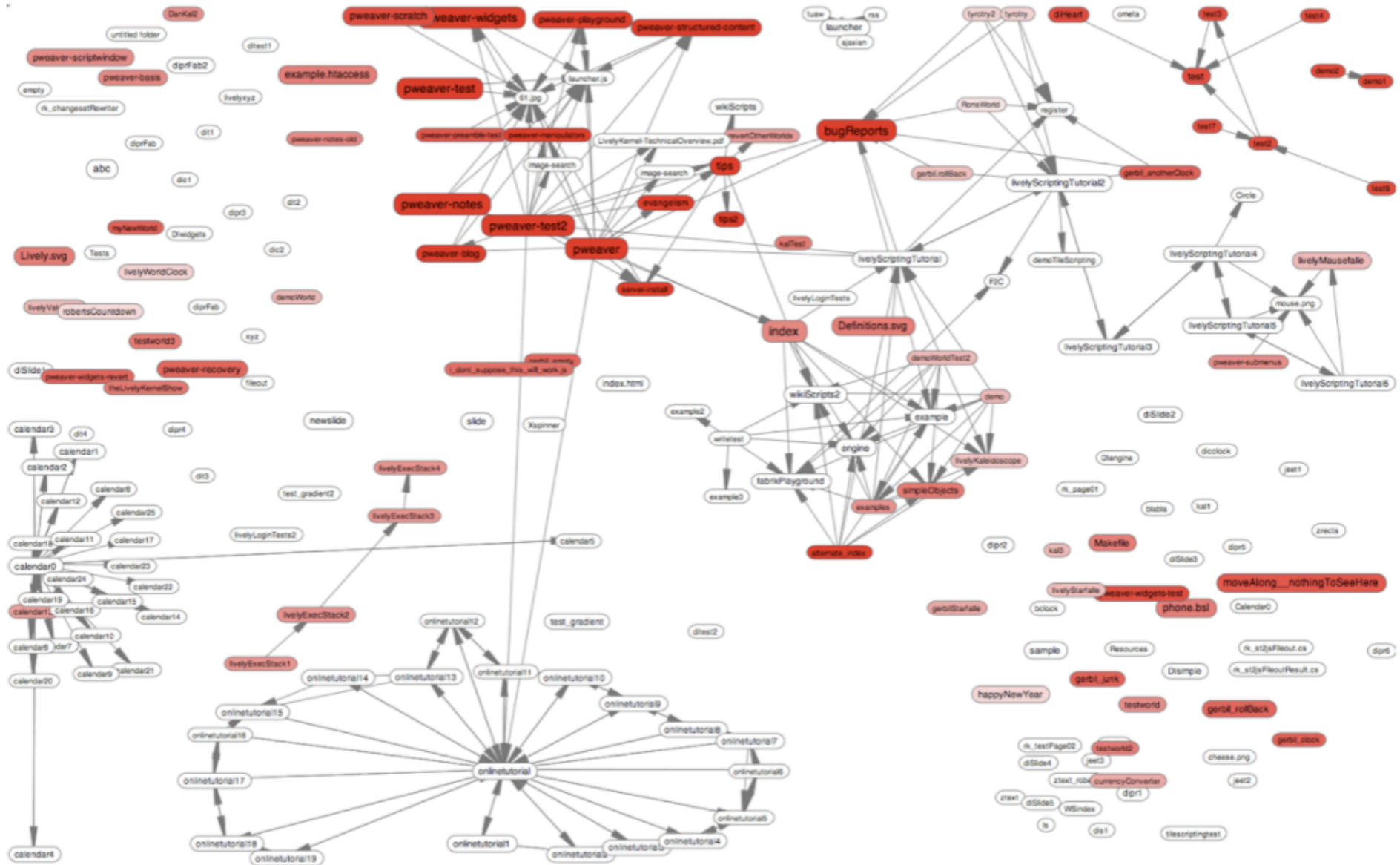
Development Tools



The screenshot displays three overlapping windows from a development environment:

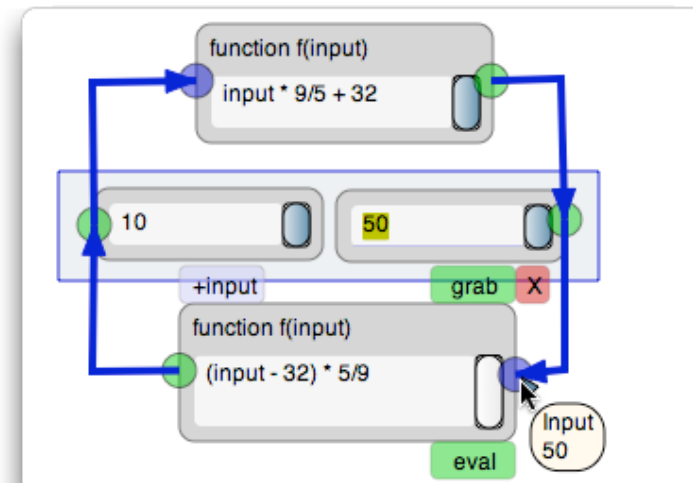
- ide.js:lively.ide.SystemBrowser**: A file browser on the left showing a list of files like Base.js, Contributions.js, Core.js, etc. Below it is a class browser with tabs for 'classes', 'functions', and 'objects'. The 'classes' tab is active, showing a tree of classes including `lively.ide.SystemBrowser`, which is highlighted. Below the tree is a code editor showing the source code for `lively.ide.SystemBrowser`, including methods like `documentation`, `initialize`, and `rootNode`.
- TestRunner**: A window on the right displaying a list of test results with execution times, such as `AComponentCopierTest (1547ms)` and `AFabrikSerializationTest (2476ms)`. At the bottom, it shows a summary: `Tests run: 413 -- Tests failed: 15 -- Time: 33.414s`. Below the summary, several test failures are listed, including `-->morph overlaps!` and `-->assert failed`.

Example: Network Visualization in Lively Wiki



Summary

- Web based end-user programming environment
- Lively Fabrik
 - Combine visual data flow with scripting
 - Create UI and behavior in one view
- Lively Wiki
 - Wiki principles: open, incremental, organic
 - Developing *inside the Web for the Web!*



SATTtoSE 2009

End-user Web Development Lively Fabrik and Lively Wiki

Hasso-Plattner-Institut Potsdam

Software Architecture Group

Prof. Dr. Robert Hirschfeld

Jens Lincke

Robert Krahn

<http://www.hpi.uni-potsdam.de/swa/>