



# Like It Or Not – Web Applications and Mashups Will Be Hot

**Tommi Mikkonen**  
Tampere University of Technology  
[tommi.mikkonen@tut.fi](mailto:tommi.mikkonen@tut.fi)

**Antero Taivalaari**  
Sun Microsystems Laboratories  
[antero.taivalaari@sun.com](mailto:antero.taivalaari@sun.com)



**Sun Labs**



# Background

- History of computing and software development is full of disruptive periods and paradigm shifts.
- The computing industry reinvents itself every 10-15 years.
- Examples of disruptive eras:
  - > Minicomputers in the 1970s
  - > Personal computers in the 1980s
  - > Object technology in the late 1980s and early 1990s
  - > Mobile devices, mobile software and Web 1.0 in the late 1990s

# The Next Paradigm Shift!

- The widespread adoption of the World Wide Web is reshaping our world in various ways.
  - > Documents, photos, music, videos, news and various other artifacts and services have already started migrating to the Web.
  - > Many industries (e.g., publishing and entertainment) are currently undergoing dramatic transformations.
- The software industry is on the brink of a similar transformation, or a paradigm shift.
  - > End-user software is moving to the Web.

# Evolution of the Web

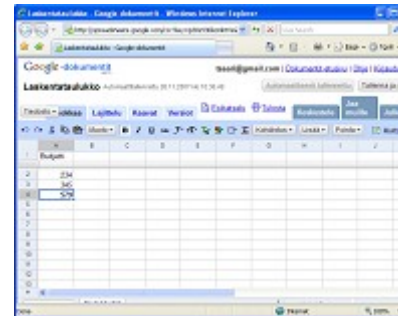


1) Simple pages with text and static images only  
(e.g., <http://www.google.com>)



2) Animated pages with plug-ins  
(e.g., <http://www.cadillac.com>)

3) Rich Internet Applications  
(e.g., [docs.google.com](http://docs.google.com))



“Mashware”

# Web Applications – Implications

- Web-based software will dramatically change the way people develop, deploy and use software.
- No more installations!
  - > Applications will simply run off the Web.
- No more upgrades!
  - > Always run the latest application version.
- Instant worldwide deployment!
  - > No middlemen or distributors needed.
- No CPU dependencies, OS dependencies, ...
  - > The Web is the Platform.

# Unfortunately...

- The web browser was not designed for running real applications.
  - > It was designed in the early 1990s for viewing documents, forms and other page-structured artifacts – *not* applications.
  - > Programming capabilities on the Web were an afterthought, not something inherent in the design of the browser.
- There is currently an *impedance mismatch* between the way software engineers have been taught to develop applications and the way the web browser requires them to be written.

# Web Development vs. Conventional Software

## Impedance Mismatch

Web Development	Conventional SW Development
<ul style="list-style-type: none"> <li>- Documents</li> <li>- Page / form oriented interaction</li> <li>- Managed graphics, static layout</li> <li>- Instant worldwide deployment</li> <li>- Source code and text favored</li> <li>- Development based mostly on conventions and “folklore”</li> <li>- Informal development practices</li> <li>- Target environment not designed for applications</li> <li>- Tool-driven development approach</li> </ul>	<ul style="list-style-type: none"> <li>- Applications</li> <li>- Direct manipulation</li> <li>- Direct, dynamic graphics</li> <li>- Conventional deployment</li> <li>- Binary representations favored</li> <li>- Development based on established engineering principles</li> <li>- More formal development</li> <li>- Target environment specifically intended for applications</li> <li>- A wide variety of development approaches available</li> </ul>

# Best Known RIA Technologies

- At this point, the following Rich Internet Application development systems are best known:
  - > Ajax
  - > Adobe AIR (Apollo)
  - > Google Web Toolkit & Google Gears
  - > Microsoft Silverlight
  - > Ruby on Rails
  - > Sun JavaFX
  - > (Sun Labs Lively Kernel)

# Landscape of RIA Technologies

## Browser-based

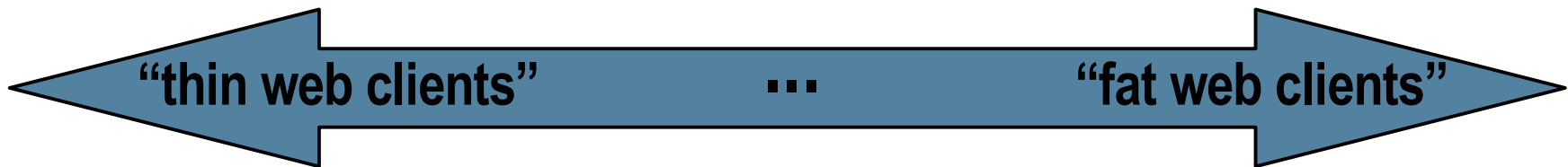
- Ajax
- Google Web Toolkit
- Sun Labs Lively Kernel

## Plugin-based

- Flash & Flex
- (Java FX, AIR)
- (Microsoft Silverlight)

## Custom runtime

- Java, Java FX
- Adobe AIR
- Silverlight



- Run in a standard browser
- No plug-ins needed
- Platform-independent
- Browser-based UI

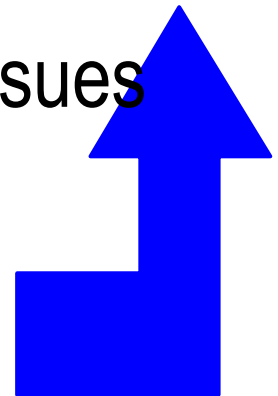
- Browser plug-in required
- Custom UI

- Custom execution engine required
- Runs outside the browser
- Custom/native UI

**Technologies in the web browser serve as the lowest common denominator!**

# Browser as a Platform – Problem Areas

- There are still a lot of problems in using the web browser as an application platform.
- Problem areas:
  - 1) Usability and user interface issues
  - 2) Networking and security issues
  - 3) Browser interoperability and compatibility issues
  - 4) Development style and testing issues
  - 5) Deployment issues
  - 6) Performance issues
  - 7) Software engineering issues



**Read our report:**

[http://research.sun.com/  
techrep/2008/abstract-175.html](http://research.sun.com/techrep/2008/abstract-175.html)

# Beyond Rich Internet Applications

# Web 2.0 – What Is It Really About?

- *Interaction.* Bringing back some of the best qualities that desktop software had before the Web, such as direct manipulation, instant feedback, piecemeal display updates.
- *Collaboration.* Allowing the users across the planet to work with each other, and share data, applications and services over the Web, regardless of their physical location.
- *Mashups.* Being able to combine content available on the Web in novel ways.

# An Important Observation

- Web applications are ***not*** just conventional desktop applications running in the web browser.
  - > Not just word processors, spreadsheets, e-mail clients, instant messengers, ...
- The Web enables the creation of entirely new types of applications and services that combine content from other web sites dynamically.
  - > This would not have been possible with conventional shrink-wrapped applications distributed in binary form.

# Mashups

- *Mashup*: A web site that combines content from more than one source (multiple web sites) into an integrated experience.
- Mashups leverage the power of the Web to support worldwide sharing of content that would not have been easily accessible or reusable before the Web.
- In principle, the content to be combined can be anything (text, source code, maps, video, blogs, product reviews, price data, ...) as long as it can be meaningfully combined with other content.

# Mashup Examples

- Chicago Police Department crime statistics mashup (<http://chicago.everyblock.com/crime/>)
- Parking availability mashups (e.g., <http://www.parkingcarma.com/>)
- Traffic tracking and congestion mashups (e.g., <http://dartmaps.mackers.com/>)
- Real estate sales and rental mashups (e.g., <http://www.housingmaps.com/>)

# Mashup – Observations

- Today, most mashups are built around *maps*.
- However, in principle the content can be anything as long as it can be digitalized and shared over the Web.
- Mashups are usually generated dynamically with no static linking; textual representations such as HTML, XML or JSON favored.
- In principle, it would be possible to build software as a mashup as well.

# Comments on Mashup Development

- Mashup development is still an *ad hoc* activity.
  - > No established development principles.
  - > Similar comments apply to web development more generally.
- The lack of proper interface descriptions and modularity makes it difficult to combine content in a systematic fashion.
- The current security model of the web browser is poorly suited to mashup development.
  - > The *Same Origin Policy* restricts access to other web sites, see [http://en.wikipedia.org/wiki/Same\\_origin\\_policy](http://en.wikipedia.org/wiki/Same_origin_policy)

# Tools for Mashup Development

- Manual development of mashups can be extremely tedious.
- Manual mashup development requires:
  - > (1) manual “scraping” of content from different web sites,
  - > (2) coming up with an algorithm (usually JavaScript code) for combining the content,
  - > (3) visualizing the resulting information using existing web technologies.
- A lot of tools for mashup development are currently under development to simplify the process.

# Best Known Mashup Development Tools

- Google Mashup Editor
  - > <http://code.google.com/gme/>
- IBM Mashup Center
  - > <http://www.ibm.com/software/info/mashup-center/>
- Intel Mash Maker
  - > <http://mashmaker.intel.com/>
- Microsoft Popfly
  - > <http://www.popfly.com/>
- Open Mashups Studio
  - > <http://www.open-mashups.org/>
- Yahoo Pipes
  - > <http://pipes.yahoo.com/>

# Example: Yahoo Pipes

The screenshot shows the Yahoo Pipes editor interface for a pipe named 'RssHuskerPedia'. The workflow consists of several interconnected steps:

- Fetch Page:** Fetches data from the URL `http://www.huskerpedia.com/index`.
- Filter:** Filters items that match all of the following rules:
  - Block items that match all of the following
  - item.content Contains <font size="2" face=
- Loop:** Iterates over each item in the input feed.
- String Replace:** Replaces the first occurrence of `<big><b>&#149;</b>` with an empty string. The results are assigned to `item.content`.
- Regex:** Uses regular expression patterns to process the data:
  - item.title: replace `<[>]*>` with `text` (checked for global, single-line, multi-line, and case-insensitive flags).
  - item.link: replace `.*` with `http://www.huskerp` (unchecked for global, single-line, multi-line, and case-insensitive flags).
  - item.y.published: replace `.*` with `text [wired]` (unchecked for global, single-line, multi-line, and case-insensitive flags).
- Rename:** Renames the processed data:
  - item.content Copy As title
  - item.content Copy As link
  - item.content Copy As y.published
  - item.content Rename description
- Loop:** A second loop iterates over the processed data.
- Date Builder:** Builds a date from `item.y.published`. The results are assigned to `item.y.published`.
- Pipe Output:** The final output of the pipe.

The interface includes a sidebar with various source and operator options, a top navigation bar with 'Run Pipe...' and 'Save a copy' buttons, and a status bar at the bottom showing 'Debugger: none'.

# Mashup Tools – Common Characteristics

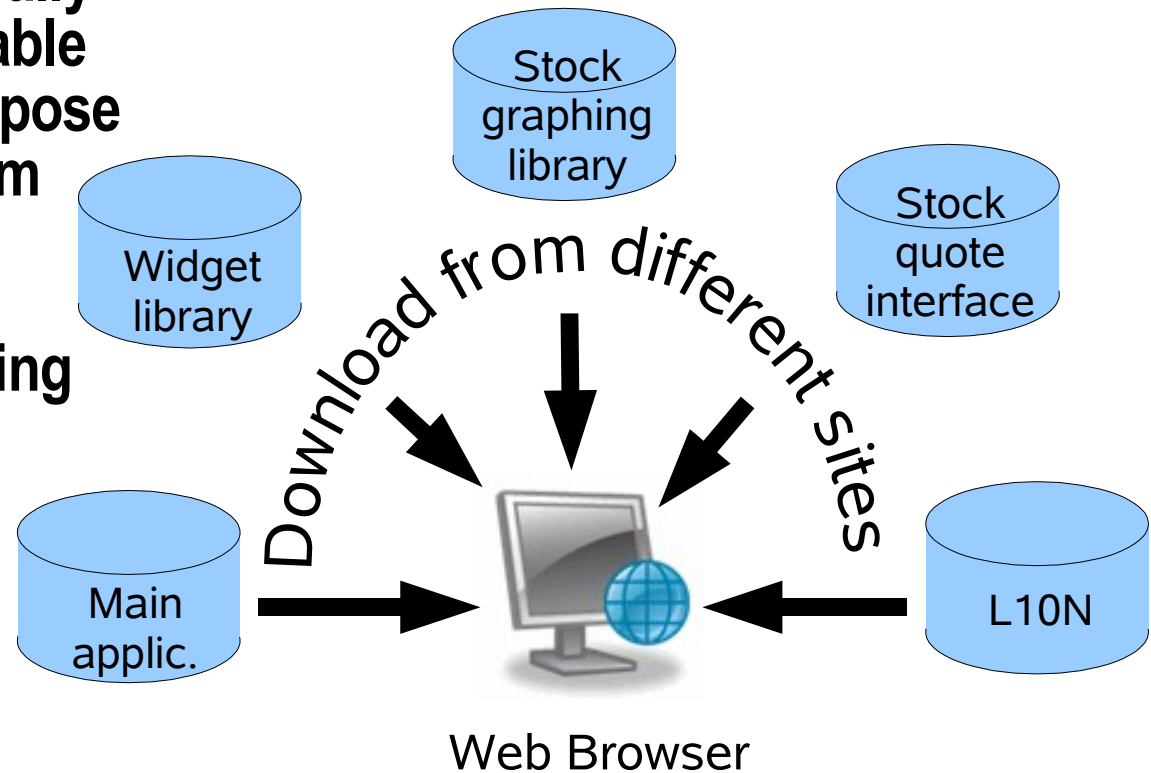
- Most of the development work occurs inside the web browser.
- Most tools offer pre-built hooks available to various existing web services (such as Google Maps, BBC Weather, Flickr...)
- Development is done visually; source code editing avoided whenever possible.
- Development intended increasingly for end users – not just professional software developers.
- Development based on dynamic languages – no compilation, no binaries, no static checking.

# Future Vision: Software as a Mashup

In the future, software will likely be built by dynamically combining the best available components for each purpose by downloading them from anywhere on the Web.

No static linking; everything downloaded on demand.

Software development will be an inherently “social” activity between developers who do not necessarily know each other.



**Today's web browsers do not support these kinds of applications yet!**

# Conclusions

- Like it or not, the Web is increasingly the platform of choice for advanced software applications.
- Web-based applications have major benefits: no installation or upgrades needed, instant worldwide deployment without middlemen.
- Web-based applications will dramatically change the way people develop, deploy and use software -> paradigm shift!
- Since the Web was not designed for applications, there are still a lot of interesting problems to solve.
- The web browser must evolve to become a better environment for applications and mashups.



**Thank You!**  
**Questions?**



**Sun Labs**

