CONFLICTS OF INTEREST
APPROACHES TO EXTENSIBLE SYSTEM DESIGN

Benjamin Lerner

April 17, 2009
The Web is Changing

Then...

- Static text
- Static links

...Now

- Multimedia
- RSS
- AJAX
- Social networks
- Dynamic content
The Browser is Changing

- **Scripting**
  - Extensive DOM (API between script and page)
  - Performance++ for interactive pages
- **New content types** (⋯)
- **New features**
  - Tabbed browsing
  - Custom toolbars
  - User scripts
- **New kinds of interactions**
  - StumbleUpon
  - Twitter
  - Remember the Milk
How to keep up?

- Ignore it: lynx
- Support new content types: 
- Support new features by hard-coding: 
- Some support limited new interactions: 
- Only one permits arbitrary extension:
Extensibility

✓ “A closed system cannot be all things to all people”
✓ Extensions customize to fit the user
✓ Add “missing” features later as extensions

✗ Extensions can cause conflicts, confusion

► How can we reason about extensions?
Goal

Systems should be designed for extensibility to accommodate adding future features. No one quite knows how to do this for the browser yet. Other system designs can help.
**Goal**

*Systems should be designed for extensibility to accommodate adding future features. No one quite knows how to do this for the browser yet. Other system designs can help.*

*Can we classify the extension models of these systems in a uniform way, and use that description to define the extension model of browsers?*
Outline

Introduction

Browser extensions
   Firefox extensions
   Sample conflicts

Classifying extensions

Applying the model
   To Firefox
   Aside: Primers on other areas
   By category

Browsers, redux
   Resolving the examples
   Conclusion
## Browsers and extensions

<table>
<thead>
<tr>
<th>Browsers</th>
<th>Operating Systems</th>
<th>Abilities</th>
</tr>
</thead>
</table>

- Browsers and their capabilities
  - Plugins: Flash, PDF, Java...
  - Device drivers
  - Arbitrary new code
  - Extensions
  - System-wide changes
  - User scripts
  - Remote thread injection
  - Limited per-app tweaks

Conflicts of Interest: Benjamin Lerner
# Browsers and extensions

<table>
<thead>
<tr>
<th>Browsers</th>
<th>Operating Systems</th>
<th>Abilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plugins: Flash, PDF, Java ...</td>
<td>Device drivers</td>
<td><em>Arbitrary new code</em></td>
</tr>
</tbody>
</table>

## Browsers
- **Browsers and extensions**
- **Browsers, redux**

## Operating Systems
- **Abilities**
- **Applies the model**
- **Classifying extensions**
- **Browser extensions**
- **Introduction**

## Abilities
- System-wide changes
- User scripts
- Remote thread injection
- Limited per-app tweaks
- Arbitrary new code

## Browsers and extensions

<table>
<thead>
<tr>
<th>Browsers</th>
<th>Operating Systems</th>
<th>Abilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plugins: Flash, PDF, Java ...</td>
<td>Device drivers</td>
<td><em>Arbitrary new code</em></td>
</tr>
</tbody>
</table>

**Conflicts of Interest**

Benjamin Lerner
# Browsers and extensions

<table>
<thead>
<tr>
<th>Browsers</th>
<th>Operating Systems</th>
<th>Abilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plugins: Flash, PDF, Java …</td>
<td>Device drivers</td>
<td>Arbitrary new code</td>
</tr>
<tr>
<td>User scripts</td>
<td>Remote thread injection</td>
<td>Limited per-app tweaks</td>
</tr>
</tbody>
</table>
## Browsers and extensions

<table>
<thead>
<tr>
<th>Browsers</th>
<th>Operating Systems</th>
<th>Abilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plugins: Flash, PDF, Java …</td>
<td>Device drivers</td>
<td>Arbitrary new code</td>
</tr>
<tr>
<td><strong>Extensions</strong></td>
<td></td>
<td>System-wide changes</td>
</tr>
<tr>
<td>User scripts</td>
<td>Remote thread</td>
<td>Limited per-app tweaks</td>
</tr>
<tr>
<td></td>
<td>injection</td>
<td></td>
</tr>
</tbody>
</table>
What can Firefox extensions do?

- **New functionality:** (just some of thousands)
  - *Web-of-Trust* rates links by “trustworthiness”
  - *Tab Preview* shows thumbnail views of tabs
  - *Session Manager* saves window/tab arrangements between sessions
What can Firefox extensions do?

- **New functionality:** (just some of thousands)
  - *Web-of-Trust* rates links by “trustworthiness”
  - *Tab Preview* shows thumbnail views of tabs
  - *Session Manager* saves window/tab arrangements between sessions

- **New policies:**
  - *AdBlock+* prevents known ads from downloading
  - *Perspectives* bypasses warnings about self-signed but stable SSL certificates
What can Firefox extensions do?

- **New functionality:** (just some of thousands)
  - *Web-of-Trust* rates links by “trustworthiness”
  - *Tab Preview* shows thumbnail views of tabs
  - *Session Manager* saves window/tab arrangements between sessions

- **New policies:**
  - *AdBlock+* prevents known ads from downloading
  - *Perspectives* bypasses warnings about self-signed but stable SSL certificates

- **New appearance:**
  - *Personas* are lightweight themes for the browser
  - *Slashdotter* is a site-specific tweaker
How do Firefox extensions work?

- Firefox defines its UI using scripts, XUL (like HTML) and CSS
- XUL defines a DOM similar to standard web programming
How do Firefox extensions work?

Extensions also define their UI using scripts, XUL, and CSS.
How do Firefox extensions work?

- Firefox and extension definitions are *merged* at load time
Resource Conflict: TabMix Plus + others

- TabMix Plus combines dozens of extensions’ abilities
- …and either disables or modifies them for compatibility
- …and does an incomplete job
Policy Conflict: FoxyTunes + FlashBlock

- FoxyTunes inserts a Flash object to stream MP3s
Policy Conflict: FoxyTunes + FlashBlock

- FoxyTunes inserts a Flash object to stream MP3s
- FlashBlock blocks all Flash

Conflicts of Interest

Benjamin Lerner
Policy Conflict: FoxyTunes + FlashBlock

- FoxyTunes inserts a Flash object to stream MP3s
- FlashBlock blocks all Flash, including FoxyTunes’
**Temporal Conflict: Linkify + Printify**

*Contrived example*
Temporal Conflict: Linkify + Printify

Test content:
This is some text that has a link in it to http://www.google.com.

* Contrived example
Temporal Conflict: Linkify + Printify

Test content:
This is some text that has a link in it to http://www.google.com.

Test content:
This is some text that has a link (http://www.google.com) in it to http://www.google.com.

Test content:
This is some text that has a link (http://www.google.com) in it to http://www.google.com.

* Contrived example
So... let’s reject all interactions?

- Overkill — *some* interactions are good
  - FireBug + Firecookie, YSlow, Jiffy: website debugger + cookies, profilers
  - Lightning + GData: calendar + Google calendar support
- ...”

- Ill-defined — *which* ones are good?
Recap

✅ Browsers merit extensions
✅ Extensions *should* interact
❌ Extensions can conflict

We need:
- a way to describe these interactions and conflicts
- to resolve conflicts
Extension model

- An extension model describes how extensions apply to a base system:
  - what they can do
  - how powerful they are
- It describes how extensions may break each other, too

A fully-designed extension model gives

✓ stable expectations for extension authors
✓ simple expectations for extension users
### The classification scheme

- **Design choices:**
  - Behaviors
  - Authorship
  - Integration time
  - Cooperation
The classification scheme

- **Design choices:**
  - Behaviors
  - Authorship
  - Integration time
  - Cooperation

- **Extension abilities:**
  - Extended resource
  - Pervasiveness
  - Granularity
  - Interactions
The classification scheme

- **Design choices:**
  - Behaviors
  - Authorship
  - Integration time
  - Cooperation

- **Extension abilities:**
  - Extended resource
  - Pervasiveness
  - Granularity
  - Interactions

- **Troubleshooting techniques:**
  - Conflicts
  - Detection
  - Prevention
The classification scheme

- **Design choices:**
  - Behaviors
  - Authorship
  - Integration time
  - Cooperation

- **Extension abilities:**
  - Extended resource
  - Pervasiveness
  - Granularity
  - Interactions

- **Troubleshooting techniques:**
  - Conflicts
  - Detection
  - Prevention
Design Choices

Cooperation:

▶ How much support must the application author build in, to support extensions fully?

Antonym: obliviousness

▶ Can the application author pretend extensions don’t exist?

Cooperation takes effort, planning

[Filman and Friedman 2000]
Extension Abilities

Pervasiveness:
- How much of the system can be affected by an extension?

Granularity:
- How small of a change is possible?

Higher pervasiveness often requires coarser granularity.
Troubleshooting Techniques

Conflict:

▶ What interactions are unwanted?

Detection:

▶ How early can conflicts be found?

\[ \text{design} \rightarrow \text{compile} \rightarrow \text{install} \rightarrow \text{load} \rightarrow \text{run} \]

Earlier conflict detection usually requires more cooperation
Firefox’s extension model

- Design choices:
  
  - Extension abilities:
    
  - Troubleshooting techniques:
Firefox’s extension model

- Design choices:
  - Heavy cooperation for XUL extension
  - Wide JS API
  - No cooperation for JS extension—it’s no need

- Extension abilities:
  - Troubleshooting techniques:
Firefox’s extension model

▶ Design choices:
  ▶ Heavy cooperation for XUL extension
  ▶ Wide JS API
  ▶ *No* cooperation for JS extension—there’s no need

▶ Extension abilities:
  ▶ Can extend individual XUL nodes—fine grained
  ▶ Can change most behaviors of system—very pervasive

▶ Troubleshooting techniques:
Firefox’s extension model

- Design choices:
  - Heavy cooperation for XUL extension
  - Wide JS API
  - No cooperation for JS extension—there’s no need

- Extension abilities:
  - Can extend individual XUL nodes—fine grained
  - Can change most behaviors of system—very pervasive

- Troubleshooting techniques:
  - None, currently
Switching gears

✔ Firefox’s extension model is powerful…

❌ …but is not easily analyzable

- Other systems have dealt with adding new code:
  - Aspect-oriented programming
  - Operating systems

- Other systems have dealt with resolving problems:
  - Feature specification
  - Security monitors

- Draw inspiration from these systems

* See paper for details
How does this apply?

Design choices

Extension abilities

Troubleshooting techniques

- Browsers
- Security monitors
- Feature specification
- Operating systems
- Aspect-oriented programming

[Denys et al. 2002; Keck and Kuehn 1998; Small and Seltzer 1996]
Primer:

Aspect-oriented programming: adding code

Key idea: In program $P$, when event $E$ occurs, take action $A$

Example: “Log the filename on each call to fopen()”

```java
class Web {
    bool showPage(...) {
        ...
        fopen("foo.html");
        ...
        if (...)
            fopen("bar.html");
        ...
        return true;
    }
}

Before every fopen
From fopen(fname)
log fname
```

[Kiczales et al. 2001]
Primer:

Aspect-oriented programming: adding code

Key idea: In program $P$, when event $E$ occurs, take action $A$
Example: “Log the filename on each call to fopen()”

Mainline

```c
class Web {
    bool showPage(...) {
        ...
        fopen("foo.html");
        ...
        if (...) 
            fopen("bar.html");
        ...
        return true;
    }
}
```

Aspect

Before every fopen
From fopen(fname)
    log fname

[Kiczales et al. 2001]
Aspect-oriented programming: adding code

Key idea: In program $P$, when event $E$ occurs, take action $A$
Example: “Log the filename on each call to fopen()”

Primer:

```c
class Web {
    bool showPage(...) {
        ...
        log "foo.html"
        fopen("foo.html");
        ...
        if (...) 
            log "bar.html"
            fopen("bar.html");
        ...
        return true;
    }
}
```

[Kiczales et al. 2001]
Primer:

Operating systems: controlling extensions

Key idea: In system $S$, expose some additional resource $R$
Example: “Add a transactional memory subsystem”

UNIX emulator

File System

SCSI Disk

Virtual Memory

SPIN Kernel

UNIX Application

UNIX system calls

[Saito and Bershad 1998]
Primer:

Operating systems: controlling extensions

Key idea: In system $S$, expose some additional resource $R$

Example: “Add a transactional memory subsystem”
Primer:
Feature specification: finding problems

Key idea: Check for conflicts in model of program
Example: “How should call forwarding work?”

Basic phone calls:

\[
\text{ALWAYS}(\text{calls}(a,b) \rightarrow \text{EVENTUALLY}(\text{talk}(a,b) \lor \text{hangup}(a)))
\]

[Felty and Namjoshi 2003]
Primer:

Feature specification: finding problems

Key idea: Check for conflicts in model of program
Example: “How should call forwarding work?”

Basic phone calls:

\[
\text{ALWAYS}(\text{calls}(a, b) \rightarrow \text{EVENTUALLY}(\text{talk}(a, b) \lor \text{hangup}(a)))
\]

Call forwarding:

\[
\text{ALWAYS}(\text{calls}(a, b) \rightarrow \text{EVENTUALLY}(\text{forward}(a, b, c) \lor \text{hangup}(a)))
\]

[Felty and Namjoshi 2003]
Primer:

Feature specification: finding problems

Key idea: Check for conflicts in model of program

Example: “How should call forwarding work?”

Basic phone calls:

\[
\text{ALWAYS}(\text{calls}(a, b) \rightarrow \text{EVENTUALLY}(\text{talk}(a, b) \lor \text{hangup}(a)))
\]

Call forwarding:

\[
\text{ALWAYS}(\text{calls}(a, b) \rightarrow \text{EVENTUALLY}(\text{forward}(a, b, c) \lor \text{hangup}(a)))
\]

▶ Is there a conflict?
▶ How to resolve it?
▶ How to indicate call-forwarding should “win”?

[Felty and Namjoshi 2003]
Primer:

Security monitors: resolving problems

Key idea: Ensure a program obeys all policies at all times
Example: “Block all pictures in spam in downloadMsg()”

Mainline

```java
class Mail {
    Msg downloadMsg(...) {
        bin = getIMAPMsg(...);
        msg = decodeIMAP(bin);

        for each (attach in msg) {
            bin = get(attach);
            msg.append(decode(bin));
        }

        return msg;
    }
}
```

Policy

```
Before every get
From get(attach)
block attach if image in spam
```
Primer: Security monitors: resolving problems

Key idea: Ensure a program obeys all policies at all times
Example: “Block all pictures in spam in downloadMsg()”

Mainline

```java
class Mail {
    Msg downloadMsg(...) {
        bin = getIMAPMsg(...);
        msg = decodeIMAP(bin);

        for each (attach in msg) {
            bin = get(attach);
            msg.append(decode(bin));
        }
        return msg;
    }
}
```

Policy

Before every get From get(attach)
block attach if image in spam
Primer:

Security monitors: resolving problems

Key idea: Ensure a program obeys all policies at all times

Example: “Block all pictures in spam in `downloadMsg()`”

```java
class Mail {
    Msg downloadMsg(...) {
        bin = getIMAPMsg(...);
        msg = decodeIMAP(bin);
        spam = Check.isSpam(msg);
        for each (attach in msg) {
            if (spam && Check.isImg(attach)) next;
            bin = get(attach);
            msg.append(decode(bin));
        }
        return msg;
    }
}
```

Before every `get`

From `get(attach)`

block `attach` if `image in spam`
Extension abilities (1):

Idea: Require more cooperation...

- for granularity and pervasiveness:
  - Let mainline authors declare extension points

```java
class Web {
    bool showPage(...) {
        ...
        fopen("foo.html");
        ...
        if (...) {
            fopen("bar.html");
            ...
            return true;
        }
    }
}
```

[Dantas and Walker 2006]
Extension abilities (1):

Idea: Require more cooperation...

▶ for granularity and pervasiveness:
  ▶ Let mainline authors declare extension points

```c
class Web {
    bool showPage(...) {
        ...
        fopen("foo.html");
        ...
        if (...) 
            fopen("bar.html");
        ...
        return true;
    }
}
```

[Dantas and Walker 2006]
Extension abilities (1):

Idea: Require more cooperation…

- for granularity and pervasiveness:
  - Let mainline authors declare extension points

- for stability:
  - and give them API-like status

```java
class Web {
  bool showPage(...) {
    ...
    fopen("foo.html");
    ...
    if (...) {fopen("bar.html");
    ...
    return true;
  }
}
```

Mainline

Aspect

Before every fopen
From fopen(fname)
log fname

[Aldrich 2005]
Extension abilities (1):

Idea: Require more cooperation…

- for granularity and pervasiveness:
  - Let mainline authors declare extension points

- for stability:
  - and give them API-like status

```
class Web {
    bool showPage(...) {
        ...
        fopen("foo.html");
        ...
        if (...)
            fopen("bar.html");
        ...
        return true;
    }
}
```

[Alrich 2005]
Extension abilities (1):

Idea: Require more cooperation…

- for granularity and pervasiveness:
  - Let mainline authors declare extension points

- for stability:
  - and give them API-like status

```java
class Web {
  bool showPage(...) {
    fopen("foo.html");
    if (...) fopen("bar.html");
    return true;
  }
}

Before every fopen
From fopen(fname)
log fname

Module

```java
class Web {
  bool showPage(...) {
    joinpoint construct-Web;
    joinpoint before-showPage;
    joinpoint p;
    joinpoint after-showPage;
  }
}

[Aldrich 2005]
```

Conflicts of Interest

Benjamin Lerner
Extension abilities (2):

Idea: Sacrifice pervasiveness for modular detection

- Use opaque types for capabilities
  - E.g. a `Console.T*` gives access to a console
  - Prevents all forged inputs

[Bershad et al. 1995]
Extension abilities (2):

Idea: Sacrifice pervasiveness for modular detection

- Use opaque types for capabilities
  - E.g. a Console.T* gives access to a console
  - Prevents all forged inputs

- Use session types to define protocols
  - E.g. for a network card,
    READY: NicEvt! → ?AckEvt → READY
  - Prevents unexpected inputs
  - ...leads to feature specification

[Bershad et al. 1995]

[Hunt and Larus 2007]
Extension abilities (2):

**Idea: Sacrifice pervasiveness for modular detection**

- Use opaque types for capabilities
  - E.g. a `Console.T*` gives access to a console
  - Prevents all forged inputs

- Use session types to define protocols
  - E.g. for a network card,
    
    ```
    READY: NicEvt! → ?AckEvt → READY
    ```
  - Prevents unexpected inputs
  - ...leads to feature specification

- Explicitly declare all resources in a manifest
  - State dependencies, modifications, etc.
  - Check each manifest modularly against base program

---

Conflicts of Interest: Benjamin Lerner
Troubleshooting techniques:

Idea: Resolve conflicts...

- by letting extensions extend each other
  - “mediator extensions”

Policy 1

```
Before every get
From get(attach)
block attach if image in spam
```

Policy 2

```
Before every get
From get(attach)
accept all attach from address book
```

[Bauer et al. 2005]
Troubleshooting techniques:

Idea: Resolve conflicts...

- by letting extensions extend each other
  - “mediator extensions”

- Policy 1
  - Before every get
  - From get(attach)
  - Suggest block attach if image in spam

- Policy 2
  - Before every get
  - From get(attach)
  - Suggest accept all attach from address book

[Bauer et al. 2005]
Troubleshooting techniques:

Idea: Resolve conflicts…

- by letting extensions extend each other
  - “mediator extensions”

Policy 1

Before every get
From get(attach)
Suggest block attach if image in spam

Unless
Policy 1 UNLESS Policy 2

Policy 2

Before every get
From get(attach)
Suggest accept all attach from address book

[Bauer et al. 2005]

Conflicts of Interest
Benjamin Lerner
Troubleshooting techniques:

Idea: Resolve conflicts…

- by letting extensions extend each other
  - “mediator extensions”

- by requiring cooperation from extension authors
  - needed for extensions to extend each other

\[
\begin{align*}
\text{ALWAYS} & (\text{calls}(a, b) \rightarrow \\
 & \text{EVENTUALLY} (\text{talk}(a, b) \lor \\
 & \quad \text{hangup}(a)) \\
\text{ALWAYS} & (\text{calls}(a, b) \rightarrow \\
 & \text{EVENTUALLY} (\text{forward}(a, b, c) \lor \\
 & \quad \text{hangup}(a))
\end{align*}
\]

[Li et al. 2005]
Troubleshooting techniques:

Idea: Resolve conflicts...

- by letting extensions extend each other
  - “mediator extensions”
- by requiring cooperation from extension authors
  - needed for extensions to extend each other

\[
\text{ALWAYS}(\text{calls}(a, b) \rightarrow \\
\text{EVENTUALLY}(\text{talk}(a, b) \lor \\
\text{hangup}(a))) \text{UNLESS } X)
\]

\[
\text{ALWAYS}(\text{calls}(a, b) \rightarrow \\
\text{EVENTUALLY}(\text{forward}(a, b, c) \lor \\
\text{hangup}(a))) \text{UNLESS } Y)
\]

Add an axiom \(\text{talk}(a, b) \neq \text{forward}(a, b, c)\)

[Li et al. 2005]
TabMix Plus + others

Key problem: no declarative manifest

- Use *declarative manifests* to claim resources
- … convenient: with XPath to name XUL nodes
TabMix Plus + others

Key problem: no declarative manifest

➤ Use *declarative manifests* to claim resources
➤ …convenient: with XPath to name XUL nodes

Key problem: code composition

➤ Detect weaving conflicts with *AOP techniques*
➤ …need namespaced, typed DOM APIs
FoxyTunes + FlashBlock

Key problem: no declarative manifest
  ▶ Use *Singularity manifests* as before
FoxyTunes + FlashBlock

Key problem: no declarative manifest
▶ Use *Singularity manifests* as before

Key problem: no clear policy for FlashBlock
▶ Define an *explicit security policy*
▶ Detect conflict with FoxyTune’s actions
▶ Define *higher-order policies* to revise one or both extension
Linkify + Printify

Key problem: no temporal dependencies

- Define Printify feature connecting <a> to text
- Define Linkify feature connecting text to <a>
- Make extensions cooperate to break the cycle
Linkify + Printify

Key problem: no temporal dependencies
  ▶ Define Printify *feature* connecting `<a>` to text
  ▶ Define Linkify *feature* connecting text to `<a>`
  ▶ Make *extensions cooperate* to break the cycle

Key problem: aspect ordering
  ▶ Use *AOP techniques* to detect weaving conflicts
Conclusion

- The browser is a system deserving extensions
- Extensions can interact and conflict in odd ways

- We defined a classification scheme for extension models
- …and specialized it to the browser
- …and used other systems’ extension models to address problems in the browser