Conflicts of Interest

Approaches to Extensible System Design

Benjamin Lerner

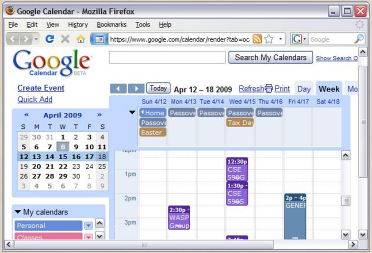
April 17, 2009

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

Browsers and extensions

Browsers	Operating Systems	*Abilities*

Browsers and extensions

Browsers	Operating Systems	*Abilities*
Plugins: Flash, PDF, Java …	Device drivers	*Arbitrary new code*

Browsers and extensions

Browsers	Operating Systems	*Abilities*
Plugins: Flash, PDF, Java …	Device drivers	*Arbitrary new code*
User scripts	Remote thread injection	*Limited per-app tweaks*

Browsers and extensions

Browsers	Operating Systems	*Abilities*
Plugins: Flash, PDF, Java …	Device drivers	*Arbitrary new code*
Extensions	???	*System-wide changes*
User scripts	Remote thread injection	*Limited per-app tweaks*

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

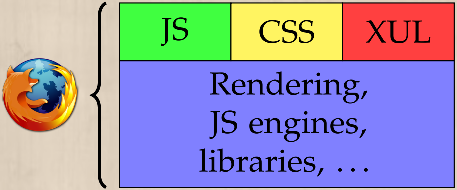
What can Firefox extensions do?

- 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?



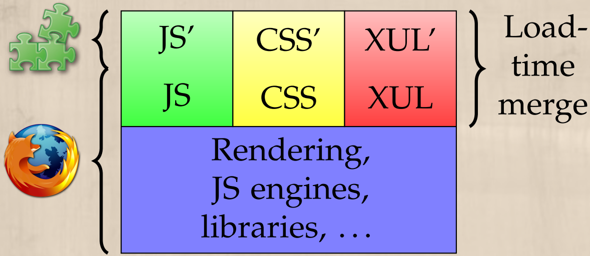
- Firefox defines its UI using scripts, XUL (like HTML) and CSS
- XUL defines a DOM similar to standard web programming

How do Firefox extensions work?

JS′	CSS′	XUL′
JS	CSS	XUL
Rendering, JS engines, libraries, …		

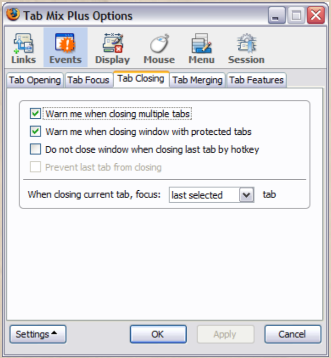
- Extensions also define their UI using scripts, XUL and CSS

How do Firefox extensions work?



- Firefox and extension definitions are *merged* at load time

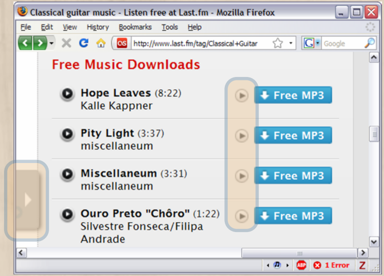
Resource Conflict: TabMix Plus + others




```
var guid_list = {};
/*
The following extensions are integrated or incompatible with Tab Mix Plus
add extensions ID in lowercase.
*/
guid_list[...] = true; // Basics
guid_list[...] = true; // BlankLast
guid_list[...] = true; // Click2Tab
guid_list[...] = true; // Close Tab On Double Click
guid_list[...] = true; // CTC
guid_list[...] = true; // Duplicate Tab
guid_list[...] = true; // Flowing Tabs
guid_list[...] = true; // FLST
guid_list[...] = true; // LastTab
guid_list[...] = true; // MiniT
guid_list[...] = true; // miniT-drag
guid_list[...] = true; // miniT-tabscroll
guid_list[...] = true; // new tab button on tab bar
guid_list[...] = true; // new tab homepage
guid_list[...] = true; // NewTabURL
guid_list[...] = true; // Petite Tabbrowser Extensions
guid_list[...] = true; // ReloadEvery
guid_list[...] = true; // Reload Tab On Double-Click
guid_list[...] = true; // Scrollable Tabs
guid_list[...] = true; // Single Window
guid_list[...] = true; // Stack style tabs
guid_list[...] = true; // superT
guid_list[...] = true; // Tab Bin
guid_list[...] = true; // Tab Clicking Options
guid_list[...] = true; // Tab Mix (original one)
guid_list[...] = true; // Tab to window
guid_list[...] = true; // Tab X
guid_list[...] = true; // Tabbrowser Extensions
guid_list[...] = true; // Tabbrowser Preferences
guid_list[...] = true; // tabdrag-for-tablib
guid_list[...] = true; // TabFX
guid_list[...] = true; // Tabs open ralative
guid_list[...] = true; // tablib
guid_list[...] = true; // Undo Close Tab
guid_list[...] = true; // Undo Close Tab
guid_list[...] = true; // Unread Tabs
```

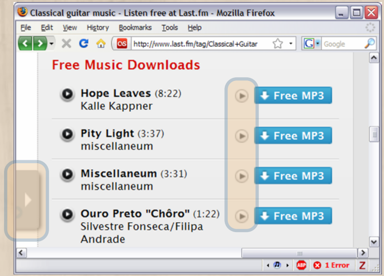
- TabMix Plus combines dozens of extensions' abilities
- …and either disables *or modifies* them for compatibility
- …and does an incomplete job

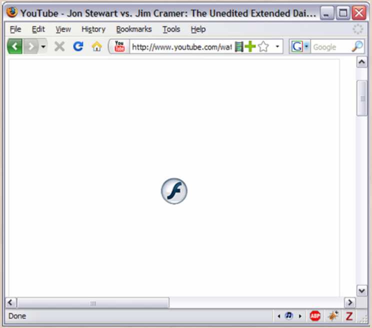
Policy Conflict: FoxyTunes + FlashBlock



- FoxyTunes inserts a Flash object to stream MP3s

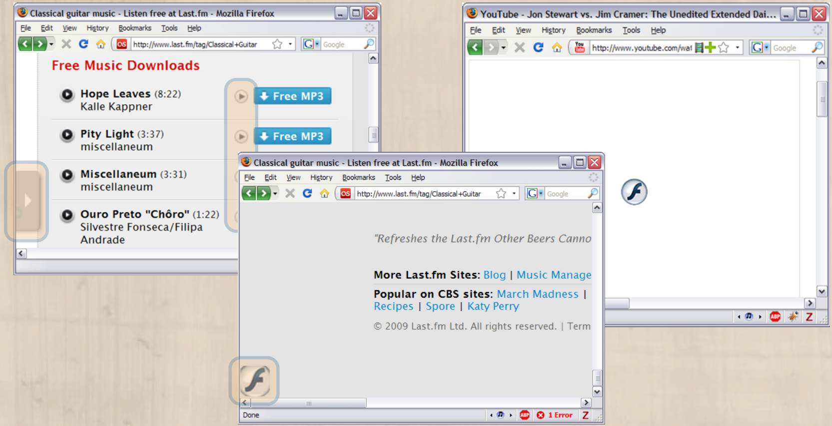
Policy Conflict: FoxyTunes + FlashBlock





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

Policy Conflict: FoxyTunes + FlashBlock

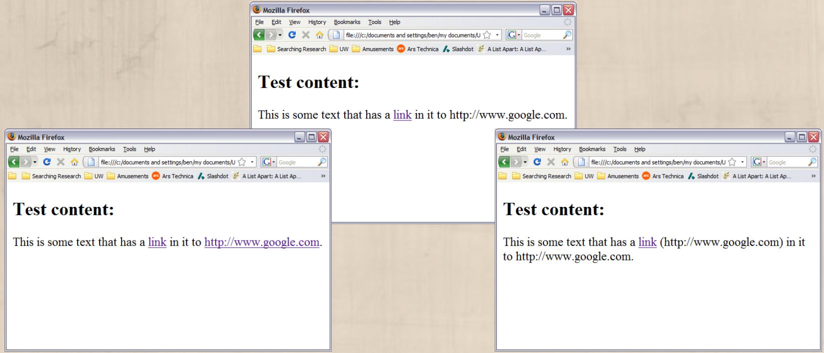


- FoxyTunes inserts a Flash object to stream MP3s
- FlashBlock blocks all Flash, including FoxyTunes'

Temporal Conflict: Linkify + Printify

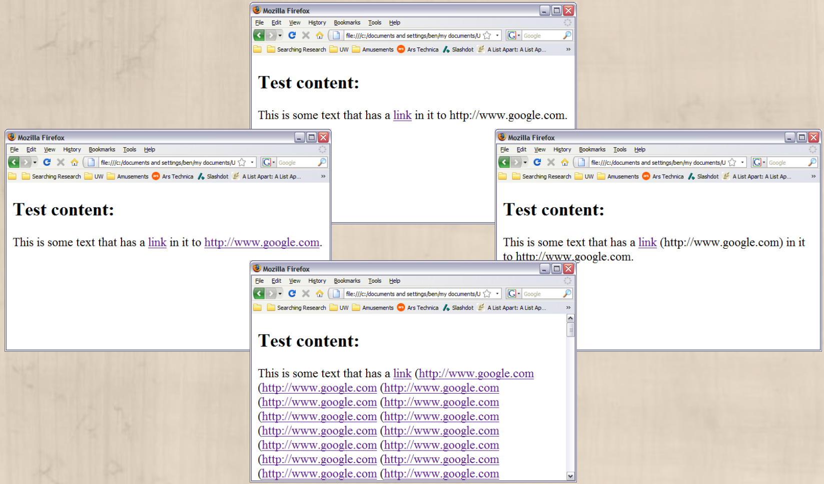
* *Contrived example*

Temporal Conflict: Linkify + Printify



** Contrived example*

Temporal Conflict: Linkify + Printify



** 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?
  design → *compile* → *install* → *load* → *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—there'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

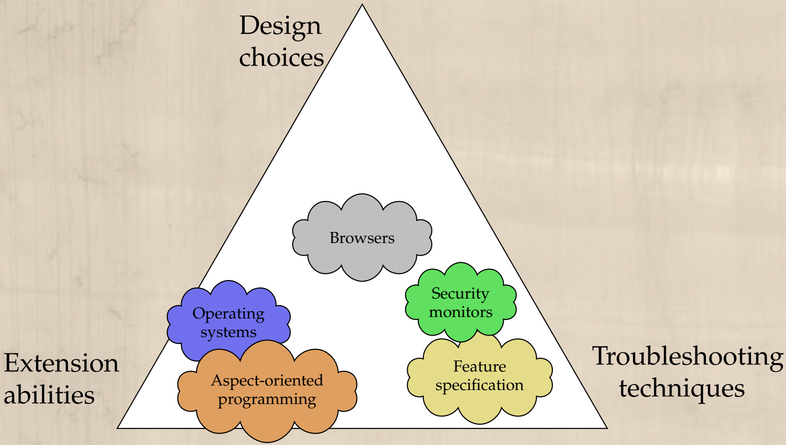
✔ Firefox's extension model is powerful…

✖ …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?



[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()`”

Mainline

```
class Web {
  bool showPage(...) {
    ...

    fopen("foo.html");
    ...
    if (...)

      fopen("bar.html");
    ...
    return true;
  }
}
```

Aspect

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

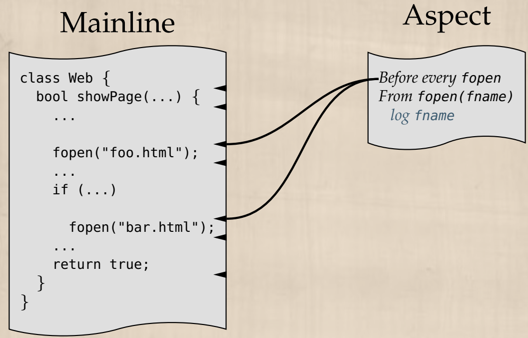
[Kiczales et al. 2001]

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()`"



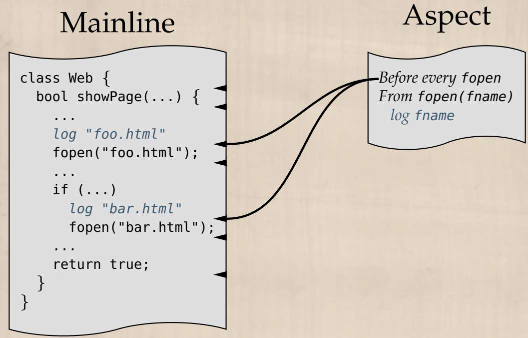
[Kiczales et al. 2001]

Primer:

Aspect-oriented programming: adding code

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

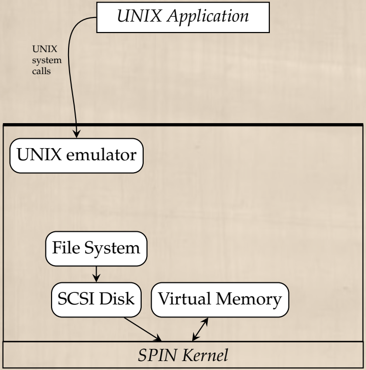
Example: "Log the filename on each call to `fopen()`"



[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”

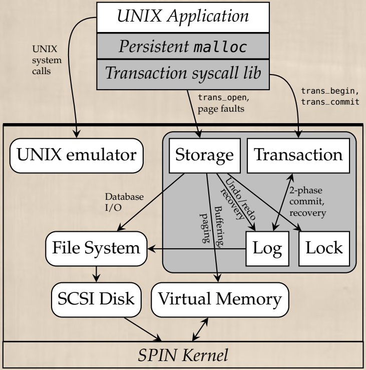


[Saito and Bershad 1998]

Primer:

Operating systems: controlling extensions

Key idea: In system *S*, expose some additional resource *R*
Example: "Add a transactional memory subsystem"



[Saito and Bershad 1998]

Primer:

Feature specification: finding problems

Key idea: Check for conflicts in model of program

Example: "How should call forwarding work?"

Basic phone calls:

$$\mathtt{ALWAYS}(calls(a,b) \rightarrow \mathtt{EVENTUALLY}(talk(a,b) \vee hangup(a)))$$

[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:

$$\texttt{ALWAYS}(\mathit{calls}(a,b) \rightarrow \texttt{EVENTUALLY}(\mathit{talk}(a,b) \vee \mathit{hangup}(a)))$$

Call forwarding:

$$\texttt{ALWAYS}(\mathit{calls}(a,b) \rightarrow \texttt{EVENTUALLY}(\mathit{forward}(a,b,c) \vee \mathit{hangup}(a)))$$

[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:

$$\texttt{ALWAYS}(calls(a, b) \rightarrow \texttt{EVENTUALLY}(talk(a, b) \vee hangup(a)))$$

Call forwarding:

$$\texttt{ALWAYS}(calls(a, b) \rightarrow \texttt{EVENTUALLY}(forward(a, b, c) \vee hangup(a)))$$

- 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

```
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

```
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

```
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;
  }
}
```

Policy

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

```
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

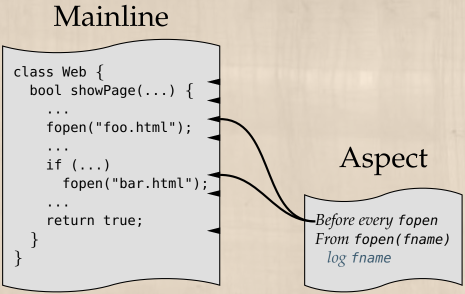
```
class Web {
  bool showPage(...) {
    ...
    fopen("foo.html");
    ...
    if (...)
      fopen("bar.html");
    ...
    return true;
  }
}
```

[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

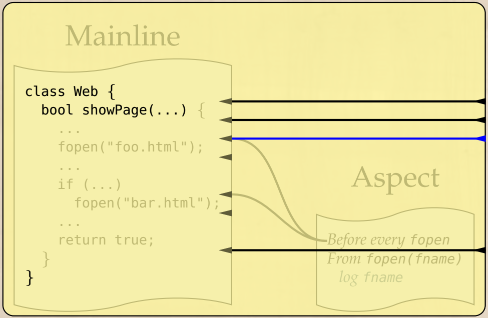


[Aldrich 2005]

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



[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

Module

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



                            joinpoint after-showPage;
}
```

[Aldrich 2005]

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

[Bershad et al. 1995]

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

[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

[Bershad et al. 1995]

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

[Hunt and Larus 2007]

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

[Plath and Ryan 2001]

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

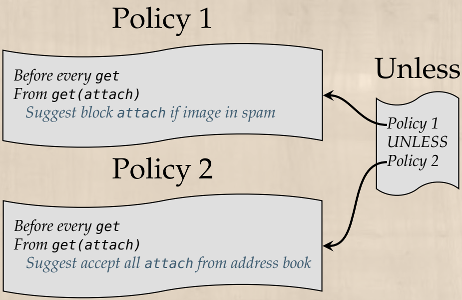
Before every `get`
From `get(attach)`
Suggest accept all `attach` *from address book*

[Bauer et al. 2005]

Troubleshooting techniques:

Idea: Resolve conflicts...

- by letting extensions extend each other
  - "mediator extensions"



[Bauer et al. 2005]

Troubleshooting techniques:

Idea: Resolve conflicts…

- by letting extensions extend each other
  - "mediator extensions"

$$\mathtt{ALWAYS}(calls(a, b) \rightarrow \mathtt{EVENTUALLY}(talk(a, b) \vee hangup(a)))$$

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

$$\mathtt{ALWAYS}(calls(a, b) \rightarrow \mathtt{EVENTUALLY}(forward(a, b, c) \vee hangup(a)))$$

[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

$$\texttt{ALWAYS}(\mathit{calls}(a,b) \rightarrow \texttt{EVENTUALLY}(\mathit{talk}(a,b) \vee \mathit{hangup}(a))\texttt{UNLESS}\ X)$$

$$\texttt{ALWAYS}(\mathit{calls}(a,b) \rightarrow \texttt{EVENTUALLY}(\mathit{forward}(a,b,c) \vee \mathit{hangup}(a))\texttt{UNLESS}\ Y)$$

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

[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