Hierarchical Layer-based Class Extensions in Squeak/Smalltalk

Matthias Springer  Hidehiko Masuhara  Robert Hirschfeld

Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology
Hasso Plattner Institute, University of Potsdam

March 14, 2016
Overview

Introduction

Examples

Mechanism

Conclusion
Introduction

- **Class Addition**: Add new method to class
- **Class Refinement**: Change (overwrite) existing method of class
- **Use Cases**
  - Convenience methods (e.g., `2.hours + 30.minutes`)
  - Bug fixing (*monkey patching*)
  - Multi-dimensional separation of concerns
    → modular understandability
  - Adding new operations to existing classes
    (c.f. *expression problem*, alternative to Visitor design pattern)
- Popular in Ruby and Smalltalk
Matriona Module System

- Module system for Squeak/Smalltalk
- Supports class nesting and class parameterization

0 Picture copyright: S. Faric, CC BY 2.0 License
Overview

Introduction

Examples

Mechanism

Conclusion
High-level Idea

Class extensions are ... 

- class members (like methods and nested classes)
- subject to *local rebinding*: avoid breaking unrelated *(black box)* code
- active in a certain scope *(locality of changes)*
  - Explicit activation: class extension specifies in which parts of the program it should be active
  - Import activation: other class *requests* a class extension (mixins)
  - Hierarchical activation: class extension is active in all nested classes
- layered: multiple class extensions can be active at the same time
- *(de)*activated similar to layers in context-oriented programming *(layer activation stack)*
Every class extension belongs to a *partial class*

Class is activated if one of its methods is executing (e.g., `Browser.open` → `Browser`)

*Scope* of class `Browser`: determines how long `Browser`’s class extensions will remain active (*deactivation* only)

*Intuition*: `scope(C)` is the set of classes which are known to be compatible with `C`’s extensions
Locality of Changes (2/3)

Example: Application calls Browser and Viewer

Browser.open(...);
Viewer.check(...);
Locality of Changes (2/3)

Example: Application calls Browser and Viewer

Application

Application
.main

Browser.open(...);
Viewer.check(...);
Locality of Changes (2/3)

Smalltalk
{Smalltalk, WebPage, Browser, Viewer, Application}

WebPage
{WebPage}
popup
open

Browser
{Browser, WebPage}
+ WebPage
popup
open
+ partial class

Viewer
{Viewer}
check

Application
{Application}
main

Example: Application calls Browser and Viewer

Application
Browser
.open
.main
Locality of Changes (2/3)

Example: Application calls Browser and Viewer

Application
.open .main
Browser
.WebPage
open
open
Viewer
check
Application
main
Browser.open(...);
Viewer.check(...);
Locality of Changes (2/3)

Example: Application calls Browser and Viewer

Application

Browser

WebView

Browser.open(...);
Viewer.check(...);
Locality of Changes (2/3)

Smalltalk \{Smalltalk, WebPage, Browser, Viewer, Application\}

- WebPage \{WebPage\}
  - popup
  - open

- Browser \{Browser, WebPage\}
  - +WebPage
  - popup
  - open

- Viewer \{Viewer\}
  - check

- Application \{Application\}
  - main

Example: Application calls Browser and Viewer

Application \{ Application \}
  - .main

Browser \{ Browser \}
  - .open

WebPage \{ WebPage \}
  - .open
  - .popup

Viewer \{ Viewer \}
  - .check

Browser.open(...);
Viewer.check(...);
Locality of Changes (2/3)

Smalltalk {Smalltalk, WebPage, Browser, Viewer, Application}

Example: Application calls Browser and Viewer

Application.open
Browser.open
WebPage.open
Viewer.check
WebPage.popup
Browser.open(...);
Viewer.check(...);

Application.main
Browser.open
WebPage.popup
Locality of Changes (2/3)

Smalltalk \{Smalltalk, WebPage, Browser, Viewer, Application\}

Example: Application calls Browser and Viewer

- Application
  - open
  - main

- Browser
  - open
  - popup

- Viewer
  - check
  - popup

- WebPage
  - open

- Application.open(...);
- Viewer.check(...);
- Browser.open(...);
- WebPage.open(...);
- WebPage.popup(...);
- Viewer.popup(...);
- WebPage.popup(...);
- Browser.WebPage.popup(...);

Partial class

Application

Browser_open

Viewer_check

WebPage_open

Browser.WebPage_popup

Example:
Application calls Browser and Viewer

Application

Browser

Viewer

WebPage

Browser.open

Viewer.check

WebPage.open

WebPage.popup

Example:
Application calls Browser and Viewer
Locality of Changes (3/3)

Example: Application calls Browser calls Viewer

```
{Smalltalk, WebPage, Browser, Viewer, Application}
```

```
+ partial class
WebPage.open(...);
Viewer.check(...);
```

```
{WebPage}
```

```
+ {Browser, WebPage}
```

```
{Viewer}
```

```
{Application}
```

```
main
```

```
WebPage.open(...);
Viewer.check(...);
```
Locality of Changes (3/3)

Example: Application calls Browser calls Viewer

Application

.main
Locality of Changes (3/3)

Example: Application calls Browser calls Viewer

```
Application.main.open();
Browser.open();
Viewer.check();
```

```
{Smalltalk, WebPage, Browser, Viewer, Application}
```

```
WebPage
popup
open
```

```
Browser
{Browser, WebPage}
```

```
Viewer
{Viewer}
```

```
Application
{Application}
```

```
WebPage.open(...);
```

```
Viewer.check(...);
```
Locality of Changes (3/3)

Example: Application calls Browser calls Viewer
Locality of Changes (3/3)

Example: Application calls Browser calls Viewer

WebPage.open(...);
Viewer.check(...);
Locality of Changes (3/3)

Example: Application calls Browser calls Viewer

Application calls Browser calls Viewer

Hierarchical Layer-based Class Extensions in Squeak/Smalltalk

Example

Application
Browser
WebPage
Viewer
Application
main

Websphere

WebPage.open(...);
Viewer.check(...);

Partial class
Locality of Changes (3/3)

Example: Application calls Browser calls Viewer
Locality of Changes (3/3)

**Smalltalk** {Smalltalk, WebPage, Browser, Viewer, Application}

- **WebPage** {WebPage}
  - `open`
  - `popup`
- **Browser** {Browser, WebPage}
  - `open`
  - `popup`
- **Viewer** {Viewer}
- **Application** {Application}

```
WebPage.open(...);
Viewer.check(...);
```

---

**Example: Application calls Browser calls Viewer**

```
Application
.open .main

Browser
.open

WebPage
.open

Viewer
.check

WebPage
.popup

Browser
.WebPage
.popup
```
Scope of a Class

Class \( L \) remains active as long as a method within \( \text{scope}(L) \) is executing.

**Definition**

\[
\text{scope}(L) = \{ L \} \cup \{ \text{target}(P) \mid P \in \text{partials}(L) \}
\]

(reflexivity)

(local rebinding)
Hierarchical Scoping

Smalltalk

AddressBook

Networking

Collections

Address

Networking

Collections

String

Pinging

add(...)
Hierarchical Scoping

- **Activation**: Add current and all enclosing classes
- **Deactivation**: For all nested classes \( N \) in \( C \): 
  \[
  \text{scope}(N) \subseteq \text{scope}(C)
  \]
Hierarchical Scoping

- **Activation:** Add current and all enclosing classes
- **Deactivation:** For all nested classes $N$ in $C$: $\text{scope}(N) \subseteq \text{scope}(C)$
Hierarchical Scoping

- **Activation**: Add current and all enclosing classes
- **Deactivation**: For all nested classes \( N \) in \( C \):
  
  \[
  \text{scope}(N) \subseteq \text{scope}(C)
  \]
Hierarchical Scoping

- **Activation**: Add current and all enclosing classes
- **Deactivation**: For all nested classes $N$ in $C$: $\text{scope}(N) \subseteq \text{scope}(C)$

Example: Address book application with remote storage

```
AddrBook
    .add
    .asAddress
String
    .asAddress
Pinging
add(...) St.AB.String.asAddress
St
AB
String
scope(St) includes
scope(AB) and scope(N)
St
AB
St
P
N
activate all
enclosing classes
Smalltalk, AddressBook

Smalltalk
{Smalltalk, AddressBook, AddressBook.Address, Networking, Networking.Address, Pinging, Collections, String}

Networking
{Networking, Address, String, Pinging}

Collections
{Collections, String}

+ partial class

AB
Network.
Pinging
St
St
P
N
```

LASSY 2016

TiTech / HPI

March 14, 2016

15 / 27
Hierarchical Layer-based Class Extensions in Squeak/Smalltalk

Examples

Hierarchical Scoping

- **Activation**: Add current and all enclosing classes
- **Deactivation**: For all nested classes \( N \) in \( C \): \( \text{scope}(N) \subseteq \text{scope}(C) \)

Example: Address book application with remote storage
Scope of a Class

Class $L$ remains active as long as a method within $\text{scope}(L)$ is executing.

Definition

$$\text{scope}(L) = \{L\} \cup \{\text{target}(P) | P \in \text{partials}(L)\}$$

$$(\text{reflexivity})$$

$$\cup \{C | C \in \text{nested}^*(\text{target}(P)) \land P \in \text{partials}(L)\}$$

$$(\text{local rebinding})$$

$$\cup \{C | C \in \text{scope}(N) \land N \in \text{nested}(L)\}$$

$$(\text{hierarch. scoping})$$
Importing Class Extensions

- Scope of a class includes scope of superclass
- Activate a class C if a method is executing in its context (polymorphic receiver class’s superclass hierarchy includes C)
Scope of a Class

Class $L$ remains active as long as a method within $\text{scope}(L)$ is executing.

Definition

$\text{scope}(L) = \{L\}$  (reflexivity)
$\cup \{C \mid C \in \text{nested}^*(\text{target}(P)) \land P \in \text{partials}(L)\}$  (loc. rebinding)
$\cup \{C \mid C \in \text{scope}(N) \land N \in \text{nested}(L)\}$  (hierarch. scoping)
$\cup \text{scope}(\text{superclass}(L))$  (importing class extensions)
Overview

Introduction

Examples

Mechanism

Conclusion
Scope of a Class

When calling or returning to a method on an object of class \( C \), activate all enclosing classes of \( C \) and \( C \).

Class \( L \) remains active as long as a method within \( \text{scope}(L) \) is executing.

**Definition**

\[
\text{scope}(L) = \{ L \} \quad \text{(reflexivity)}
\]
\[
\cup \{ C \mid C \in \text{nested}^*(\text{target}(P)) \land P \in \text{partials}(L) \} \quad \text{(loc. rebinding)}
\]
\[
\cup \{ C \mid C \in \text{scope}(N) \land N \in \text{nested}(L) \} \quad \text{(hierarch. scoping)}
\]
\[
\cup \text{scope}(	ext{superclass}(L)) \quad \text{(importing class extensions)}
\]
Effective Superclass Hierarchy

- Insert active partial classes in superclass hierarchy
- Activation order determines order of partial classes
- No `proceed` statement, use `super` instead
Overview

Introduction

Examples

Mechanism

Conclusion
Conclusion

• **High-level idea:** Make sure that class extensions are not destructive by confining their scope to *compatible* classes

• **Scoping dimensions:** explicit scoping, class nesting hierarchy (*hierarchical scoping*), superclass hierarchy (*import scoping*)

• Future Work
  – Implementation details and performance optimizations
  – Blocks/anonymous functions
  – Formal semantics of the mechanism
Appendix
Locality of Changes (4/3)

Example: Standalone usage of Viewer

WebPage .Browser is never activated
Locality of Changes (5/3)

Smalltalk
  {Smalltalk, WebPage, Browser, Viewer, Application}

WebPage
  {WebPage}

Browser
  {Browser, WebPage}

Viewer
  {Viewer}

Application
  {Application}

Example: Standalone usage of Browser

Browser.open
WebPage.open
WebPage.popup
Browser.WebPage.popup

Browser.WebPage.popup is called