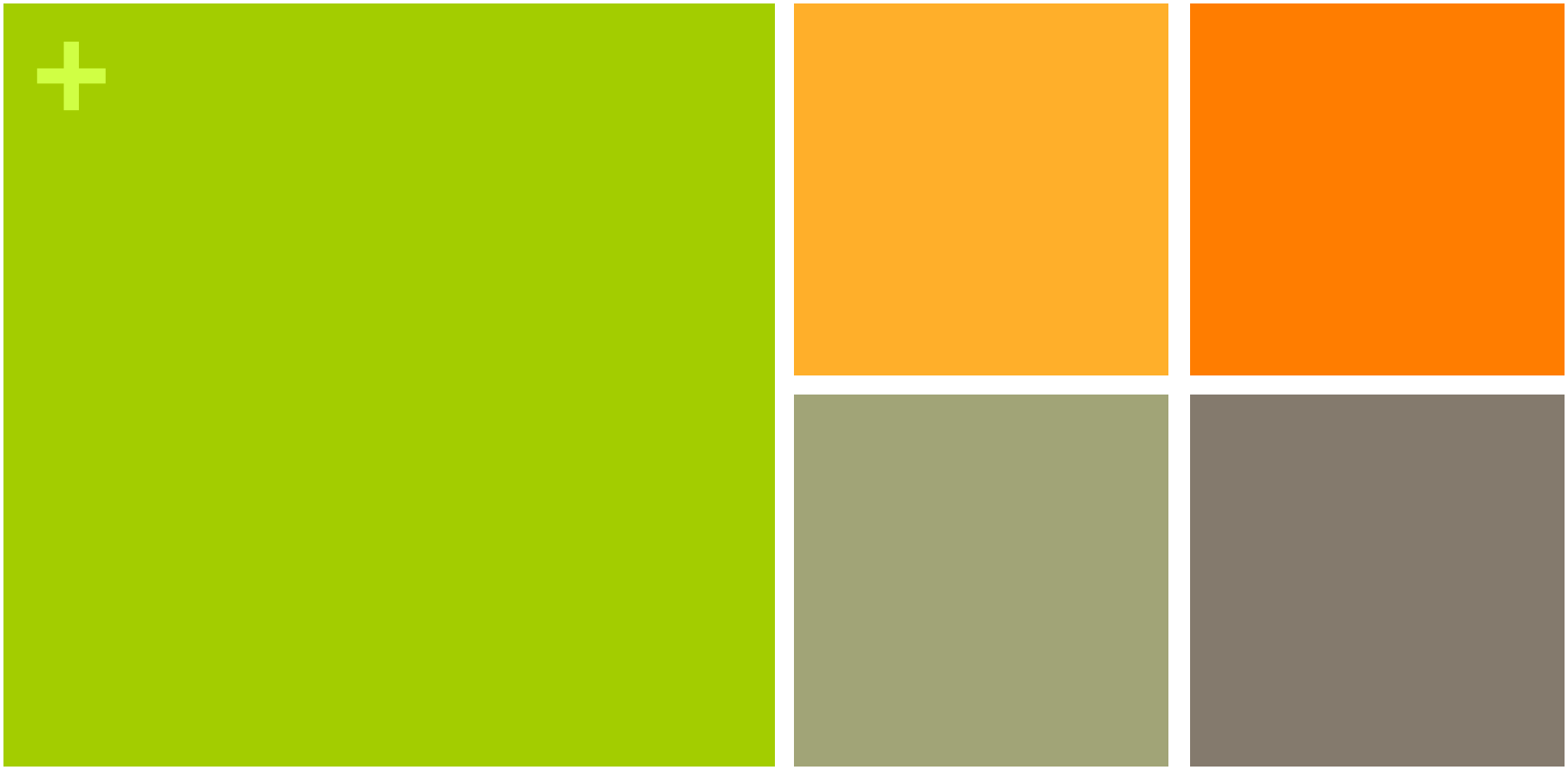
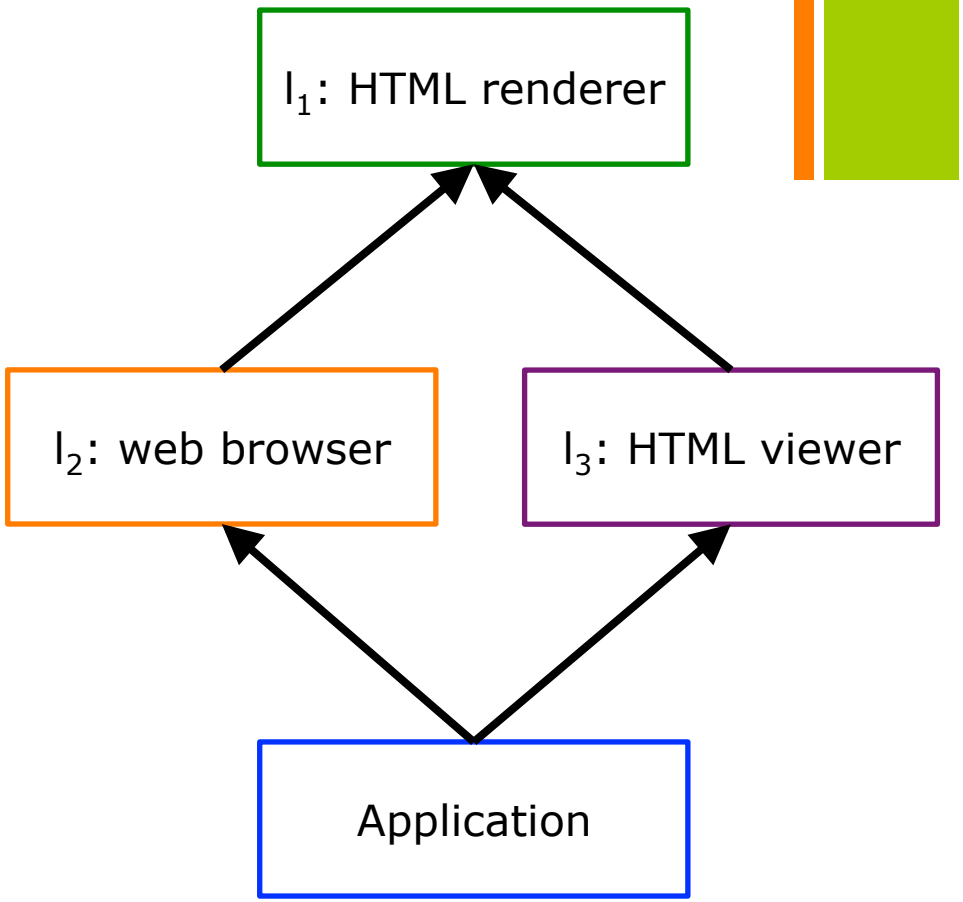


Method Shells: avoiding conflicts on destructive class extensions by implicit context switches

Wakana Takeshita and Shigeru Chiba
The University of Tokyo

Application and Software

- Modern applications are built on top of existing libraries
  - but often not as they are
- Need to modify the libraries



Mechanisms for modular customization

- Ruby's openclass
- AspectJ's aspect
  - ['01 Kiczales et al.]
- GluonJ
  - ['10 Chiba et al.]
- Classboxes
  - ['10 Bergel et al.]
- etc…

Example in GluonJ

l_1: HTML renderer

```
class Webpage{
    void popup(HTML text){
        // show a popup window
    }
    void onClick(Mouse m){
        URL url = m.getURL();
        popup(url);
        …
    }
}
```

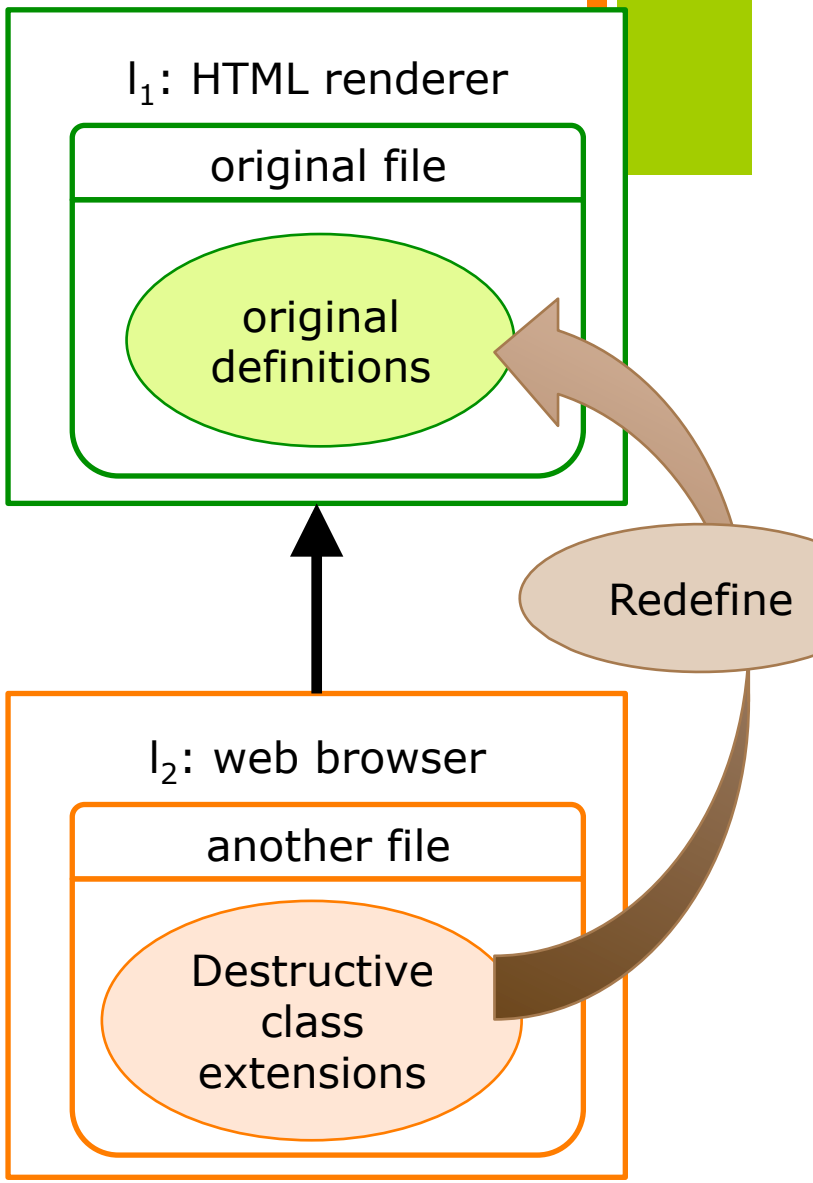
Redefine

l_2: web browser

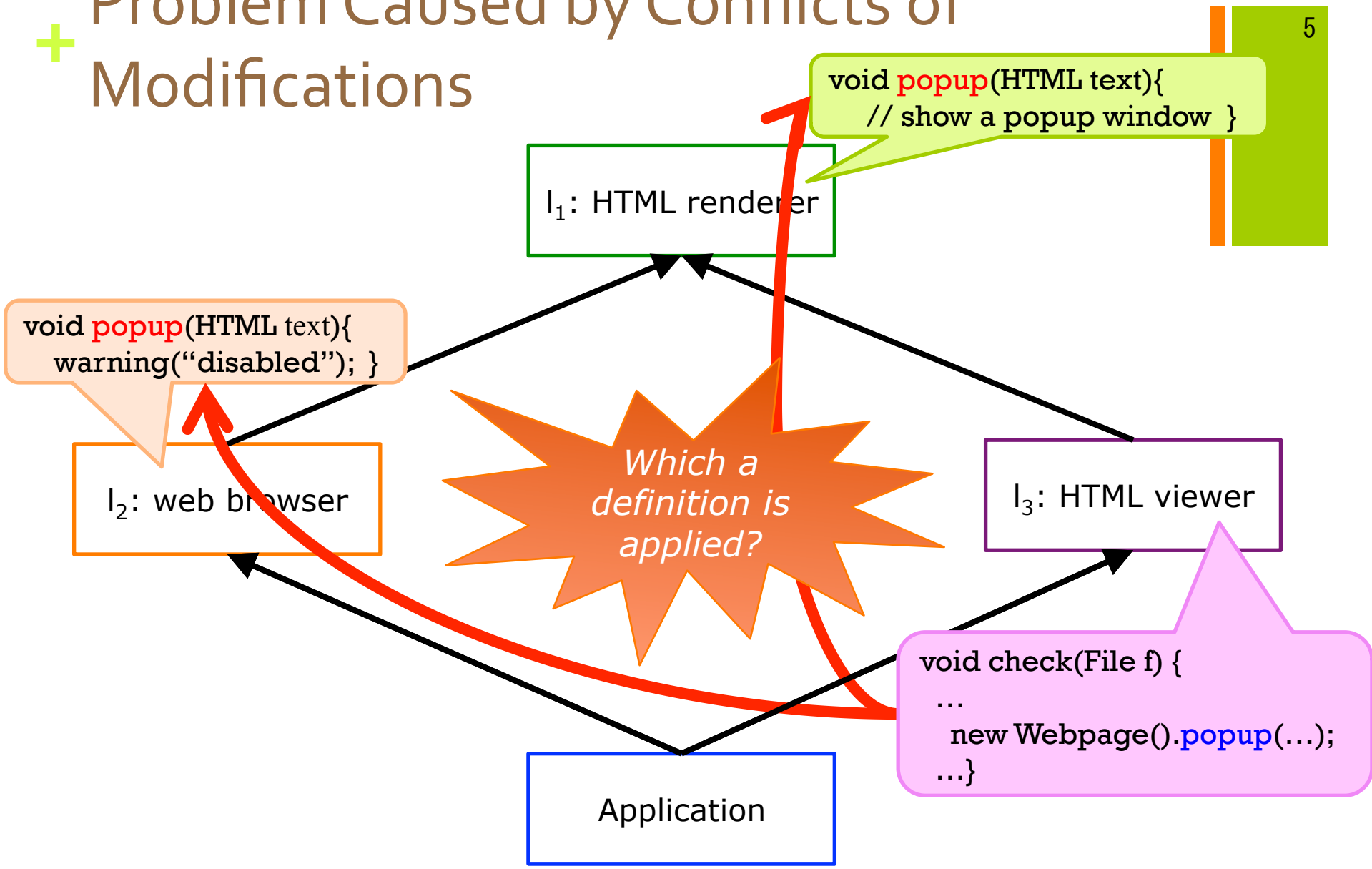
```
revise Webpage{
    void popup(HTML text){
        warning("disabled");
    }
}
```

Destructive Class Extensions

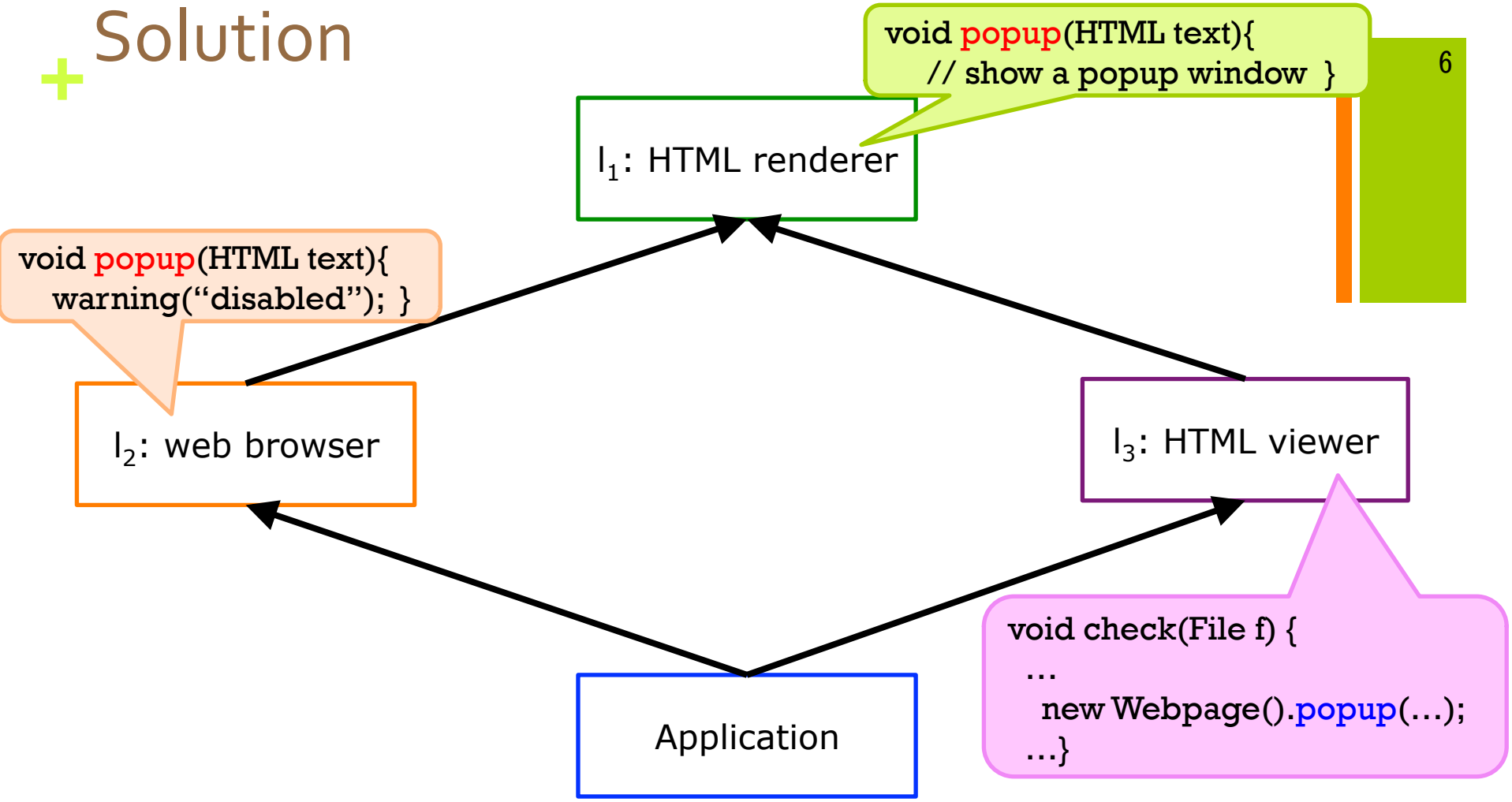
- A mechanism for writing only differences in a separate source file without rewriting existing code
  - for adding new methods to existing classes, and
  - for redefining existing methods
- Modularization of modifications
  - Dividing the work



Problem Caused by Conflicts of Modifications

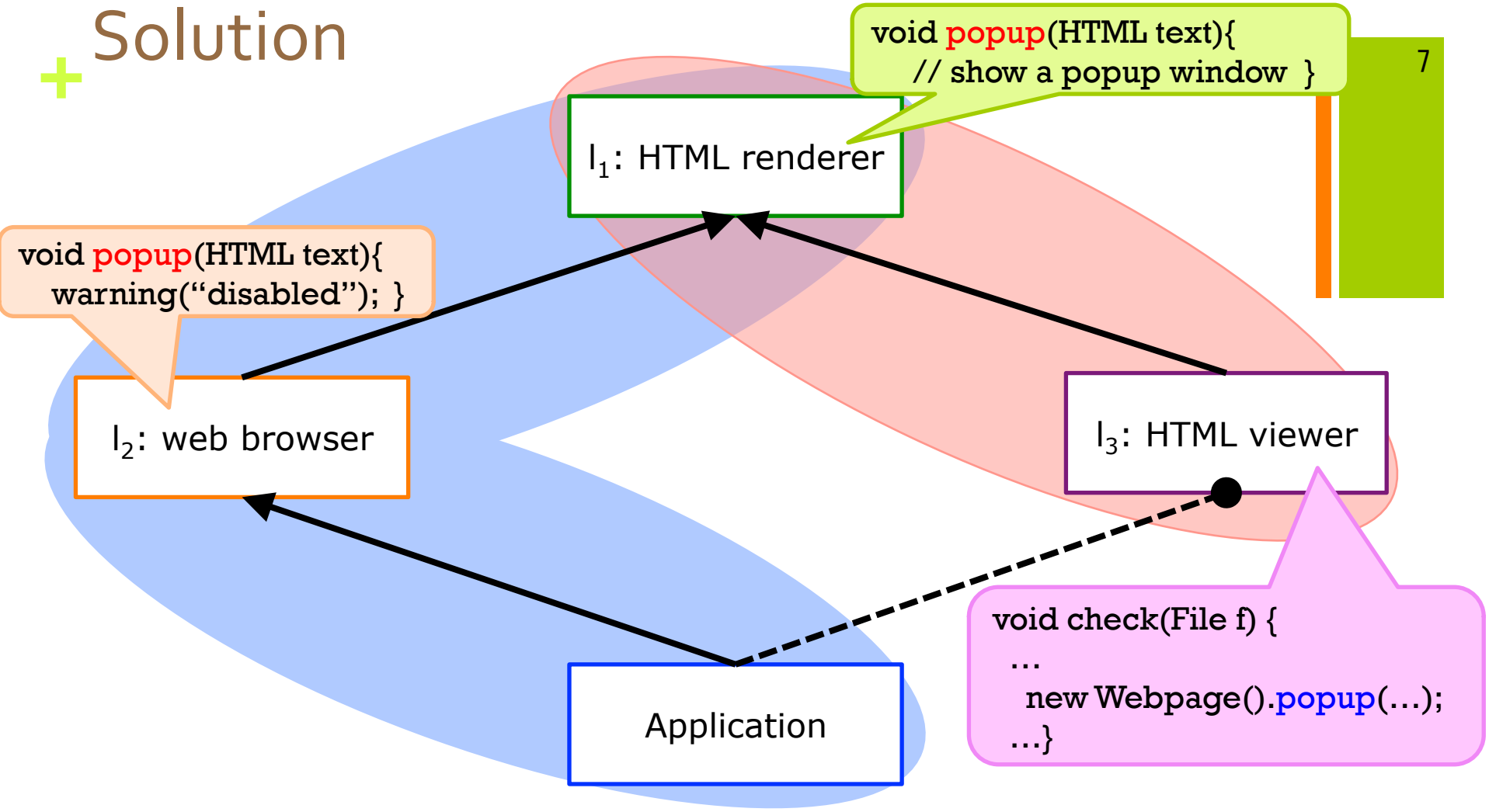


Solution



- Switch the applied destructive class extensions depending on the program's context
- How to define the context?
  - **by using two types of import**
  - **one context = a set of boxes connected by arrows**

Solution



- Switch the applied destructive class extensions depending on the program's context
- How to define the context?
  - **by using two types of import**
  - **one context = a set of boxes connected by arrows**

Proposal: Method Shells

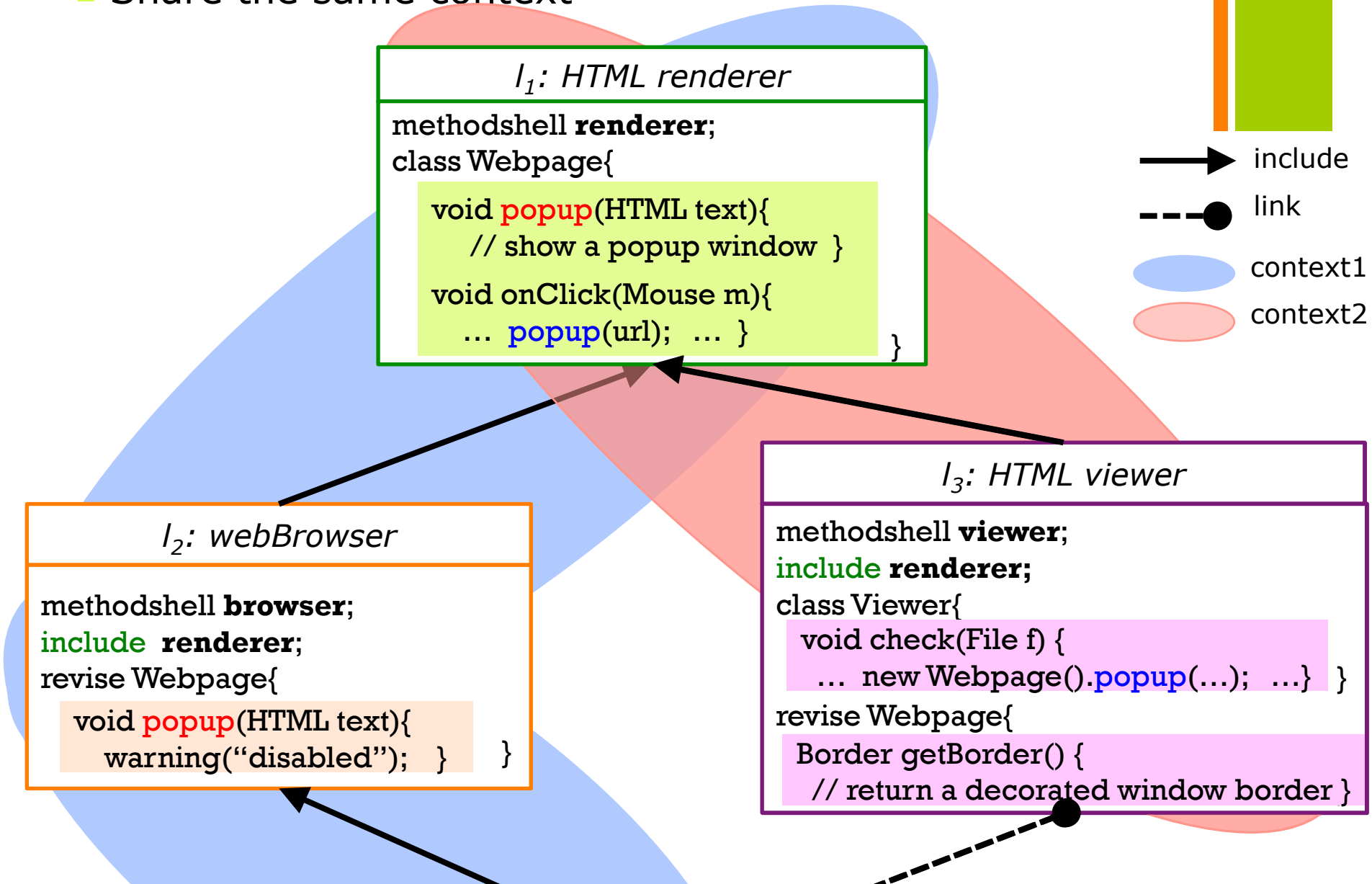
- a module system for avoiding conflicts by switching a set of revisers to fit execution contexts during runtime.
- The context is defined by two declarations
  - **Link**: for switching the contexts
  - **Include**: for sharing the
- module: methodshell
  - contains two declarations, class declarations and destructive class extensions

I_2: webBrowser

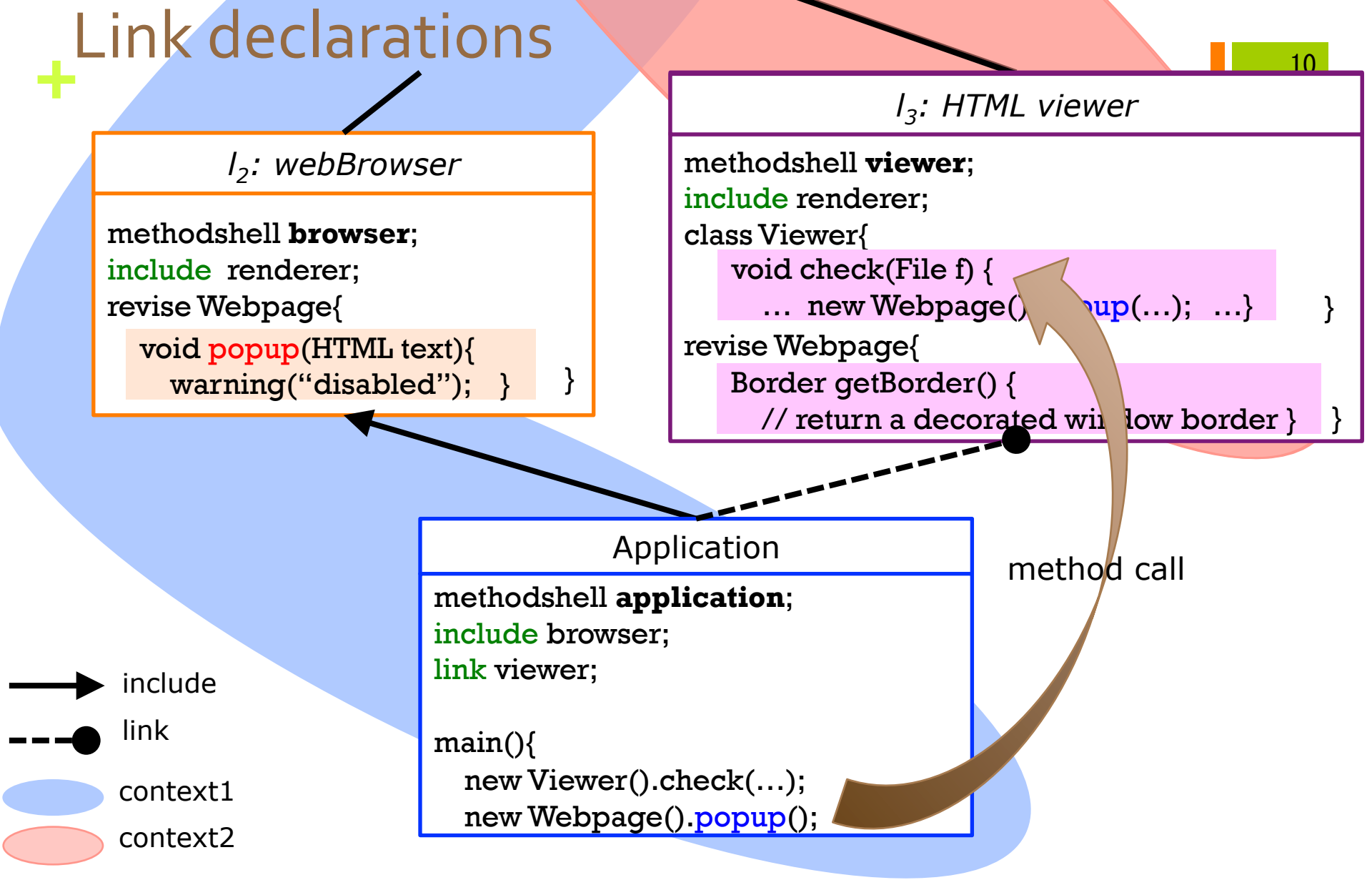
```
methodshell browser;
//include and link declarations
include  renderer;

//declarations
revise Webpage{
    void popup(HTML text){
        warning("disabled");
    }
}
```

Include declarations

- Used when to revise other methodshells
- Share the same context

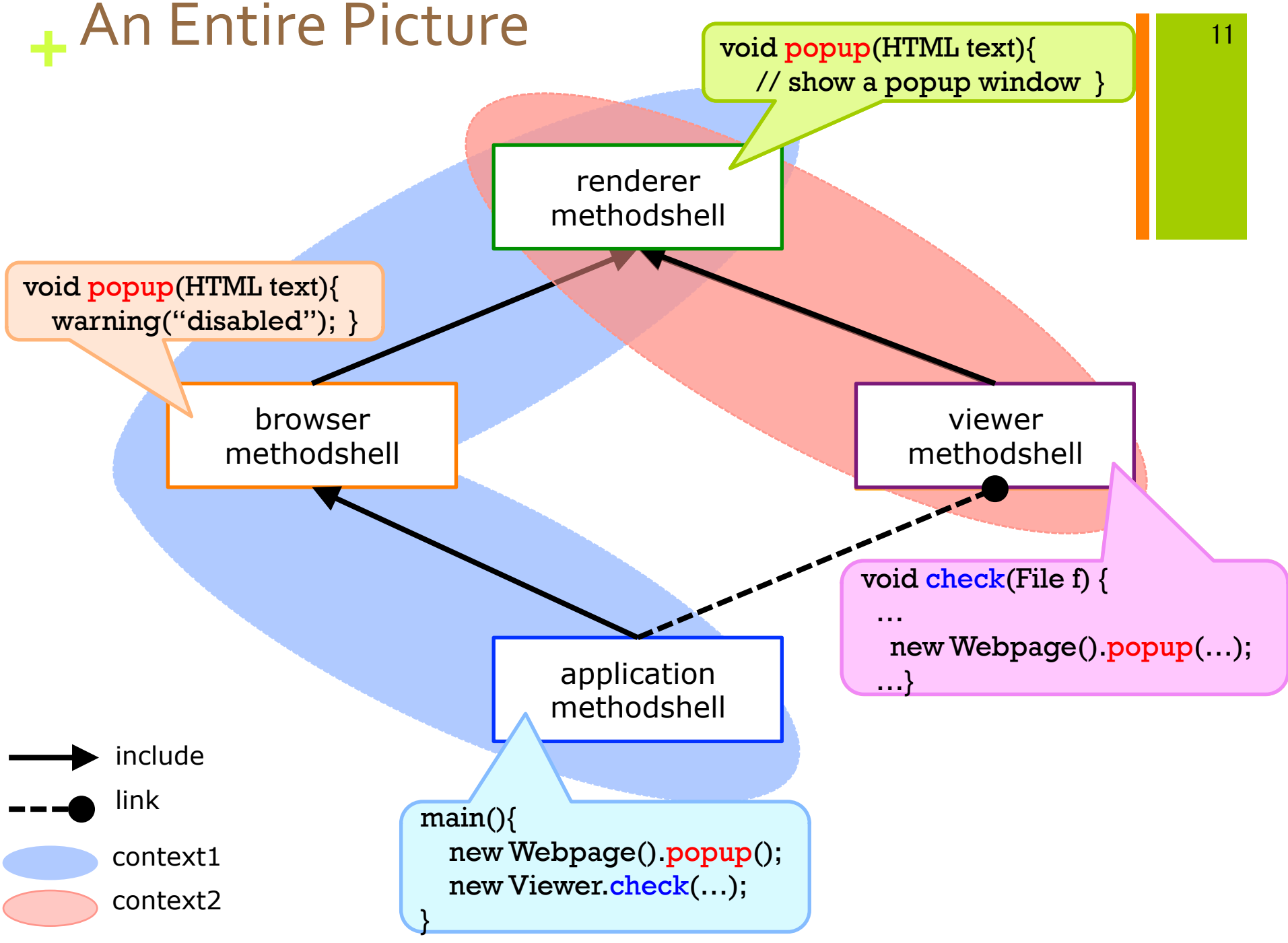


Link declarations



- When the method in the linked methodshell is called, the current context switches to the context containing the called method.

An Entire Picture



Semantics

$$\frac{\begin{array}{ll} & \overline{T_c} = \textit{linked-shells}(S) \\ \exists T_{c1} \in \overline{T_c} & \texttt{CRT(S,C)=(revise C}(\overline{M})\texttt{)||(class C---[---}\overline{M}\texttt{])} \\ & \texttt{B m}(\overline{B}\ \overline{x})\texttt{\{return e;\}}\in\overline{M} \\ \forall T_{cj} \in \overline{T_c}(i \neq j) & \texttt{C m } \textit{is not defined in } T_{cj} \end{array}}{\textit{mbody}(\texttt{m,C,S},S_c)=\overline{x}.\texttt{e in T};T_{c1}}$$

$$\frac{\begin{array}{ll} & \overline{T_c} = \textit{linked-shells}(S) \\ \forall T_{c1} \in \overline{T_c} & \texttt{C m } \textit{is not defined in } T_{cj} \\ & \textit{mbodyshell}(\texttt{m,C},S_c)=\overline{x}.\texttt{e in T} \end{array}}{\textit{mbody}(\texttt{m,C,S},S_c)=\overline{x}.\texttt{e in T};S_c}$$

$$\frac{\begin{array}{ll} & \overline{T_{c1}} = \textit{linked-shells}(S) \\ \forall T_{c1} \in \overline{T_c} & \textit{mbodyshell}(\texttt{m,C},T_{c1})=\texttt{null} \\ & \textit{mbodyshell}(\texttt{m,C},S_c)=\texttt{null} \end{array}}{\textit{mbody}(\texttt{m,C,S},S_c)=\textit{mbodyglobal}(\texttt{m,C,S},S_c)}$$

$$\frac{}{\textit{mbody}(\texttt{m,C,null,null})=\textit{mbodyglobal}(\texttt{m,C,null,null})}$$

$$\frac{\begin{array}{l} \texttt{CRT(S,C)=(revise C}(\overline{M})\texttt{)||(class C---[---}\overline{M}\texttt{])} \\ \texttt{B m}(\overline{B}\ \overline{x})\texttt{\{return e;\}}\in\overline{M} \end{array}}{\textit{mbodyshell}(\texttt{m,C,S})=\overline{x}.\texttt{e in S}}$$

$$\frac{\begin{array}{ll} & \texttt{CRT(S,C)=(revise C}(\overline{M})\texttt{)||(class C---[---}\overline{M}\texttt{])} \\ & \texttt{m is not defined in } \overline{M} \\ & \overline{S} = \textit{includings}(S) \\ \exists S_1 \in \overline{S} & \textit{mbodyshell}(\texttt{m,C},S_1)=\overline{x}.\texttt{e in T} \\ \forall S_j \in \overline{S}(i \neq j) & \textit{mbodyshell}(\texttt{m,C},S_j)=\texttt{null} \end{array}}{\textit{mbodyshell}(\texttt{m,C,S})=\overline{x}.\texttt{e in T}}$$

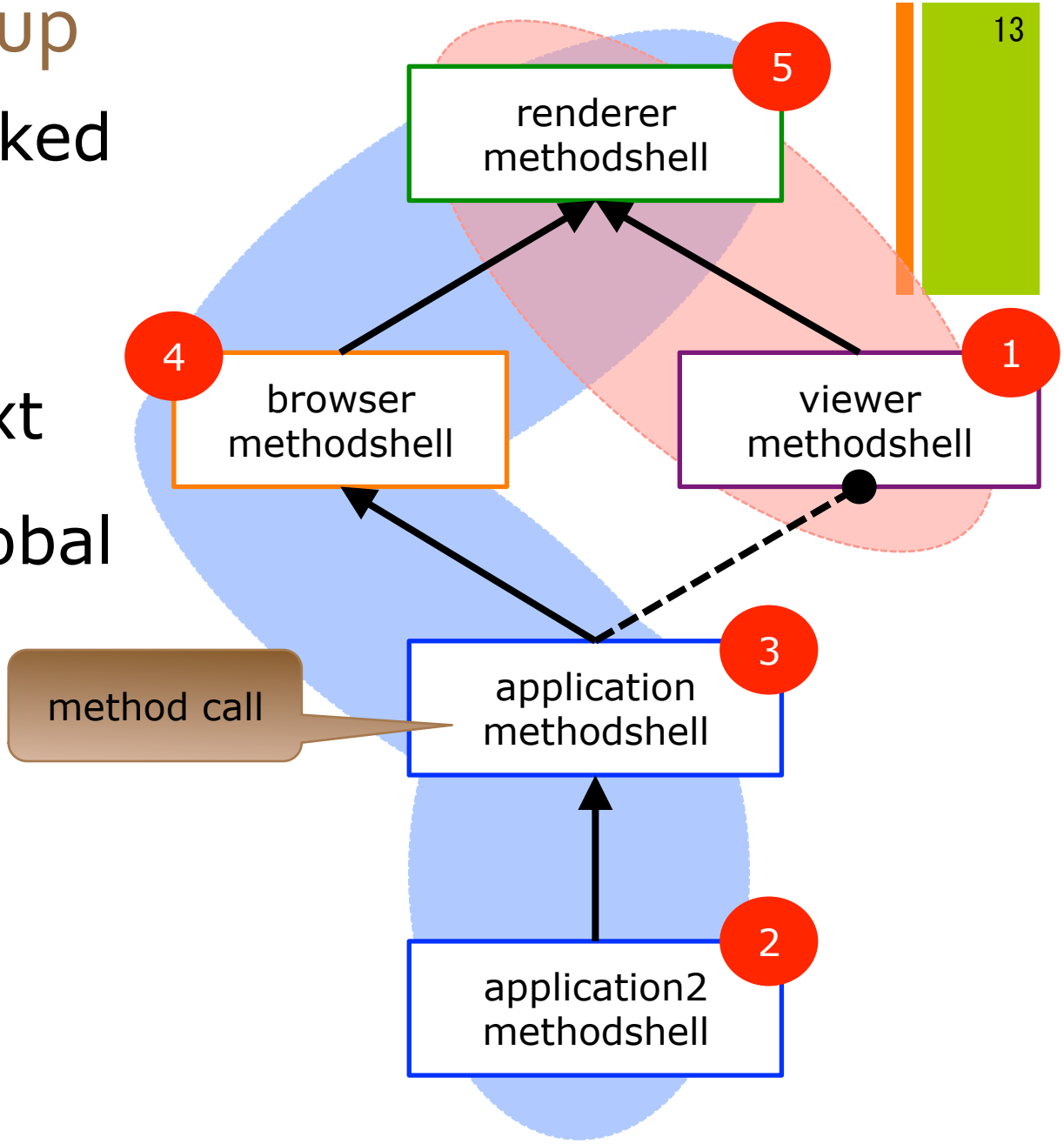
$$\frac{\begin{array}{l} \texttt{CRT(S,C)=(revise C}(\overline{M})\texttt{)||(class C---[---}\overline{M}\texttt{])} \\ \texttt{m is not defined in } \overline{M} \\ \textit{includings}(S)=\texttt{null} \end{array}}{\textit{mbodyshell}(\texttt{m,C,S})=\texttt{null}}$$

$$\frac{\begin{array}{l} \texttt{CRT(Global,C)=class C extends D\{}\overline{C}\ \overline{f}\texttt{;K }\overline{M}\texttt{\}} \\ \texttt{B m}(\overline{B}\ \overline{x})\texttt{\{return e;\}}\in\overline{M} \end{array}}{\textit{mbodyglobal}(\texttt{m,C,S},S_c)=\overline{x}.\texttt{e in null;null}}$$

$$\frac{\begin{array}{l} \texttt{CRT(Global,C)=class C extends D\{}\overline{C}\ \overline{f}\texttt{;K }\overline{M}\texttt{\}} \\ \texttt{m is not defined in } \overline{M} \end{array}}{\textit{mbodyglobal}(\texttt{m,C,S},S_c)=\textit{mbody}(\texttt{m,D,S},S_c)}$$

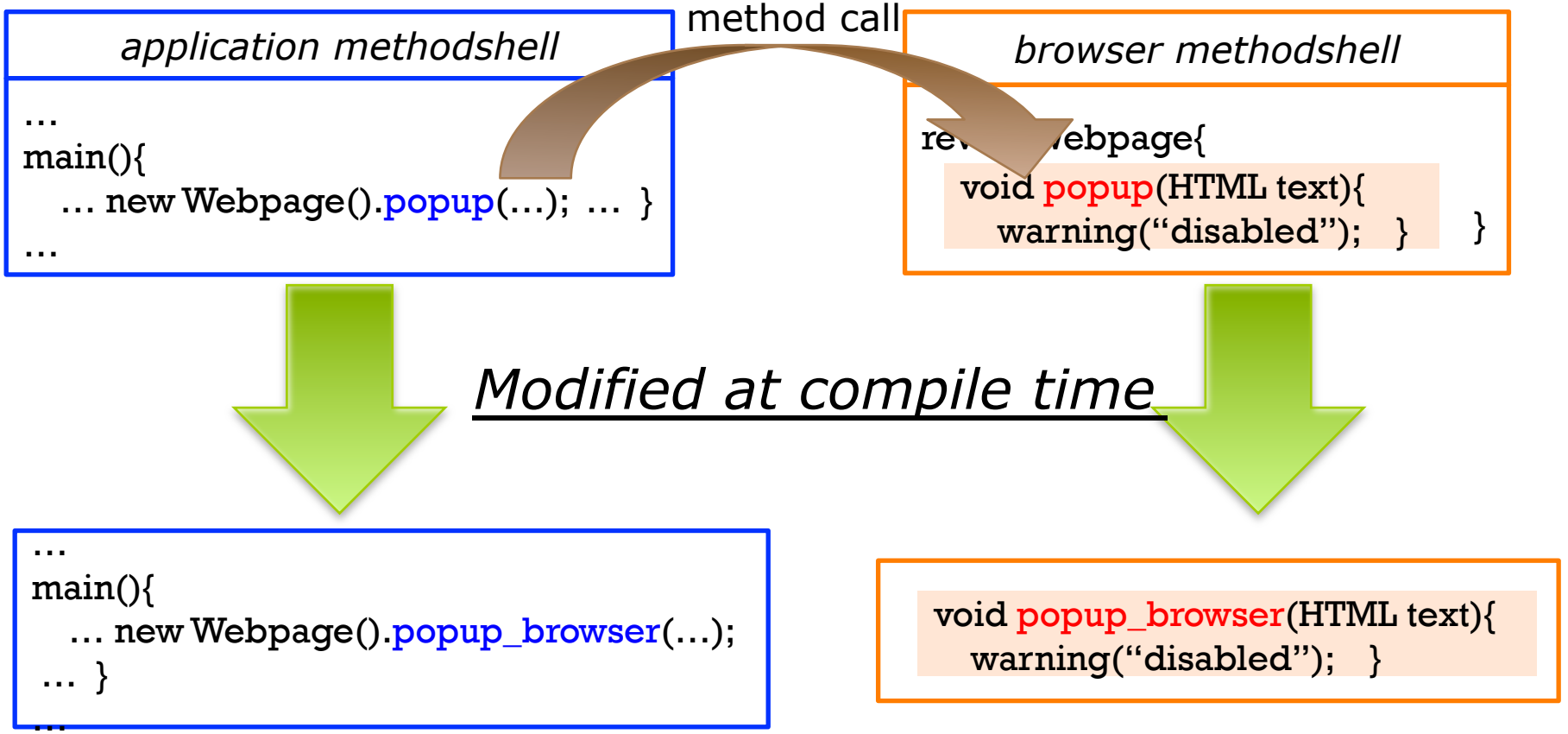
Method Lookup

1. Search the linked methodshell
2. Search the current context
3. Search the global context



A Technique for Implementation

- To reduce runtime penalty
  - by transforming unique method names and method bodies to fit them at compile time



Related Work

- Context-oriented Programming[’08 Robert H, et al.]
  - Applied method declarations are changes by the current context
- Classboxes[’05 Alexandre B, et al.]
- Ruby’s refinements
  - controlling the scope of destructive class extensions.
- NewSpeak[‘10 Bracha G, et al.]
  - All class names are virtual
  - Provide a mechanism corresponding to include declarations

Conclusion

- Demand to modify the existing code
  - There are some mechanisms that make it easy to add and redefine methods
- Proposal: Method Shells
  - a module system for switching applied definitions by context switches
  - Link and include declarations

Related Work

- NewSpeak[`10 Bracha G, et al.]
  - All class names are virtual
  - Provide a mechanism corresponding to include declarations
- Method Shelter
  - Our previous study