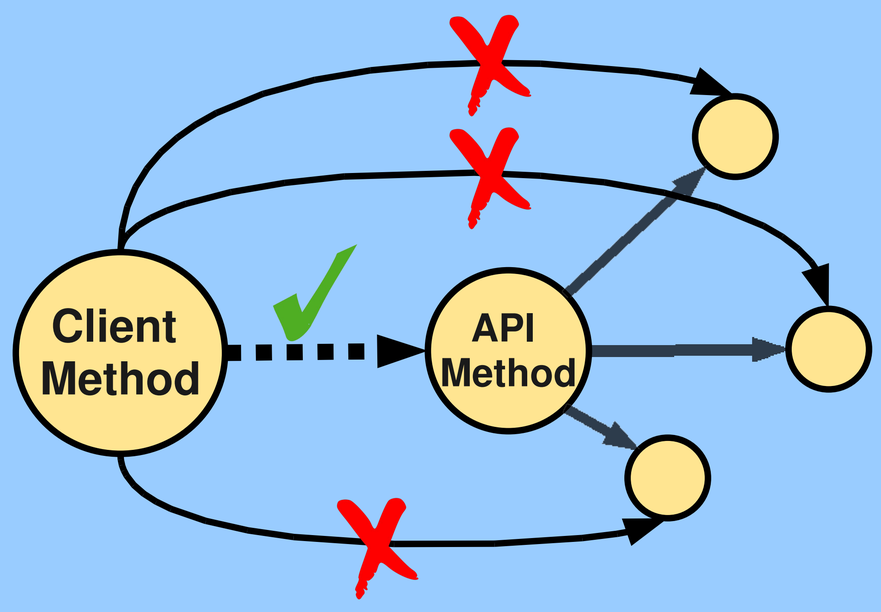
Detecting Inefficient API Usage

David Kawrykow and Martin P. Robillard – McGill University

http://swevo.cs.mcgill.ca

Good API Usage avoids Redundant Code




```
gettingURLResponse(String url) {
 return new WebConversation().getResponse(url);
}
```

vs.

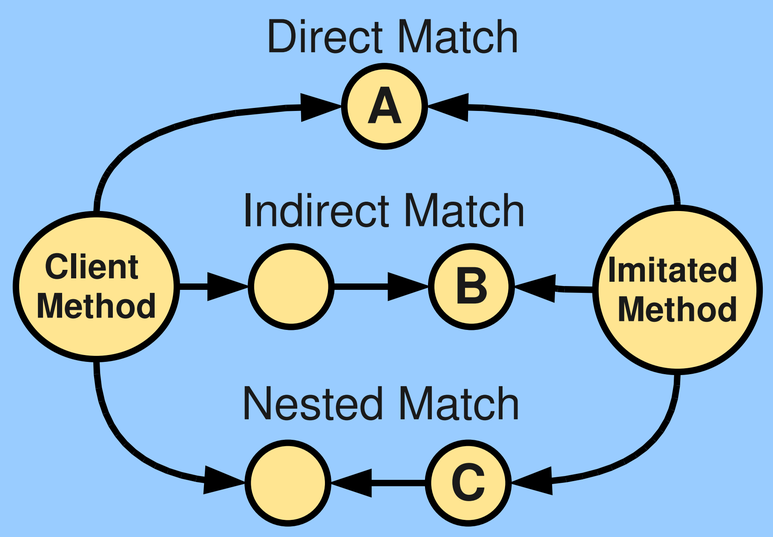
```
gettingURLResponse(String url) {
 WebResponse response = null;
 URL server = new URL(url);
 conversation = new WebConversation();
 req = new GetMethodWebRequest(server,"");
 response = conversation.getResponse(req);
 return response;
}
```

1

Reimplementing available API methods means more code maintenance.

A Novel Approach for Detecting Inefficient API Usage

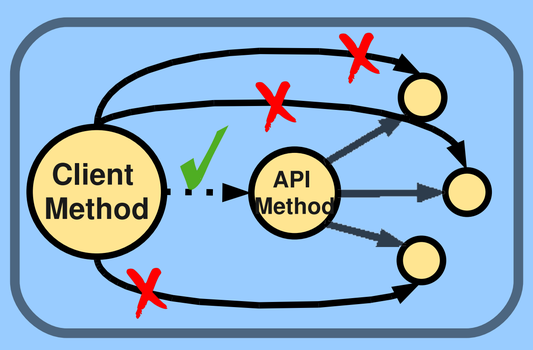
```
getResponse(String url) {
 return aWindow.getResponse(url)
}
```

2

If a client **matches** the element references of the API method, the API method is considered **imitated**. Various **filters** eliminate false positives.

Reuse Validated Imitations as API Usage Patterns



API Usage Pattern

Store

DB

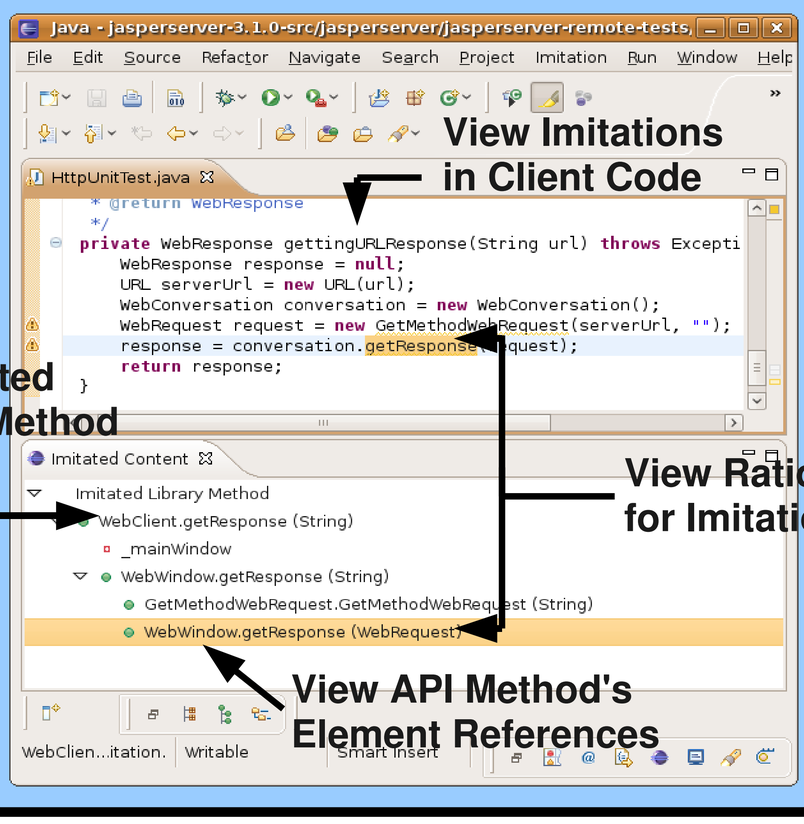
Improve Code Quality

```
gettingURLResponse(String url) {
 WebResponse response = null;
 URL server = new URL(url);
 conversation = new WebConversation();
 req = new GetMethodWebRequest(server,"");
 response = conversation.getResponse(req);
 return response;
}
```

Your Project

3

Tool Support



4