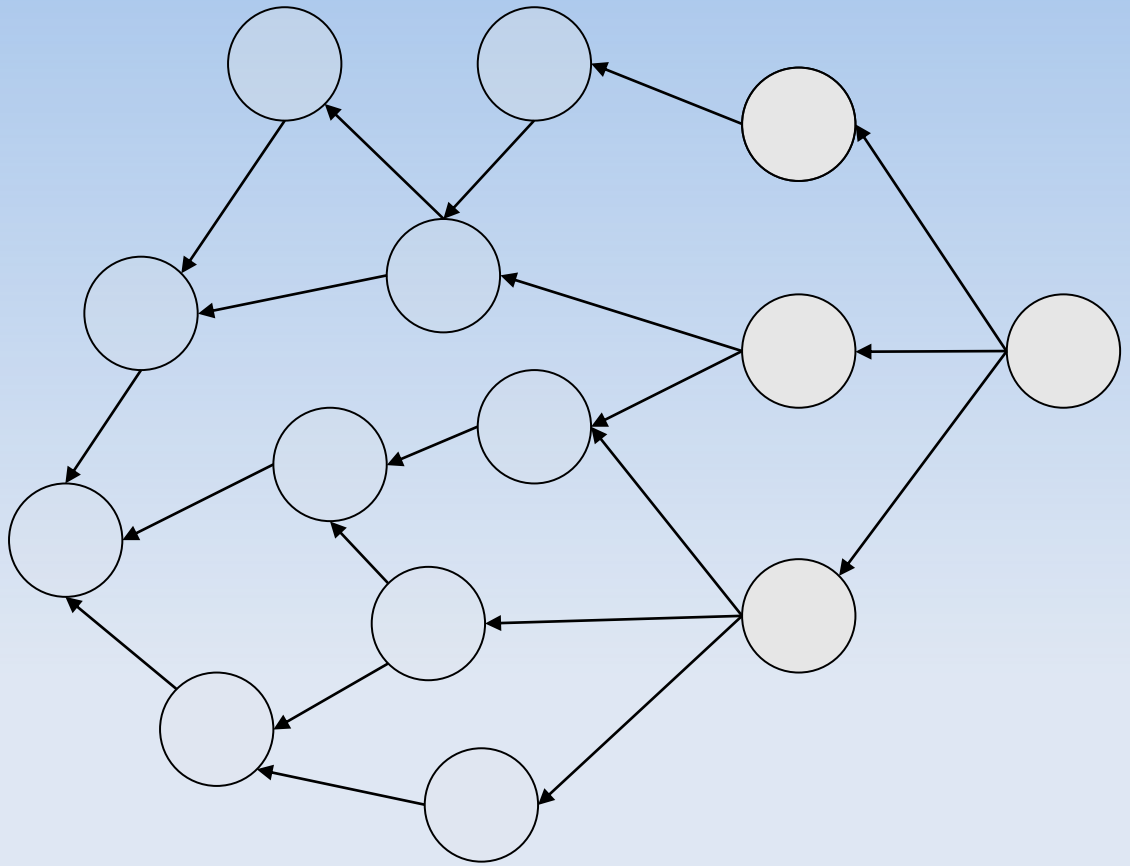
Detecting Inefficient API Usage

David Kawrykow and Martin Robillard
McGill University

Application Programming Interfaces



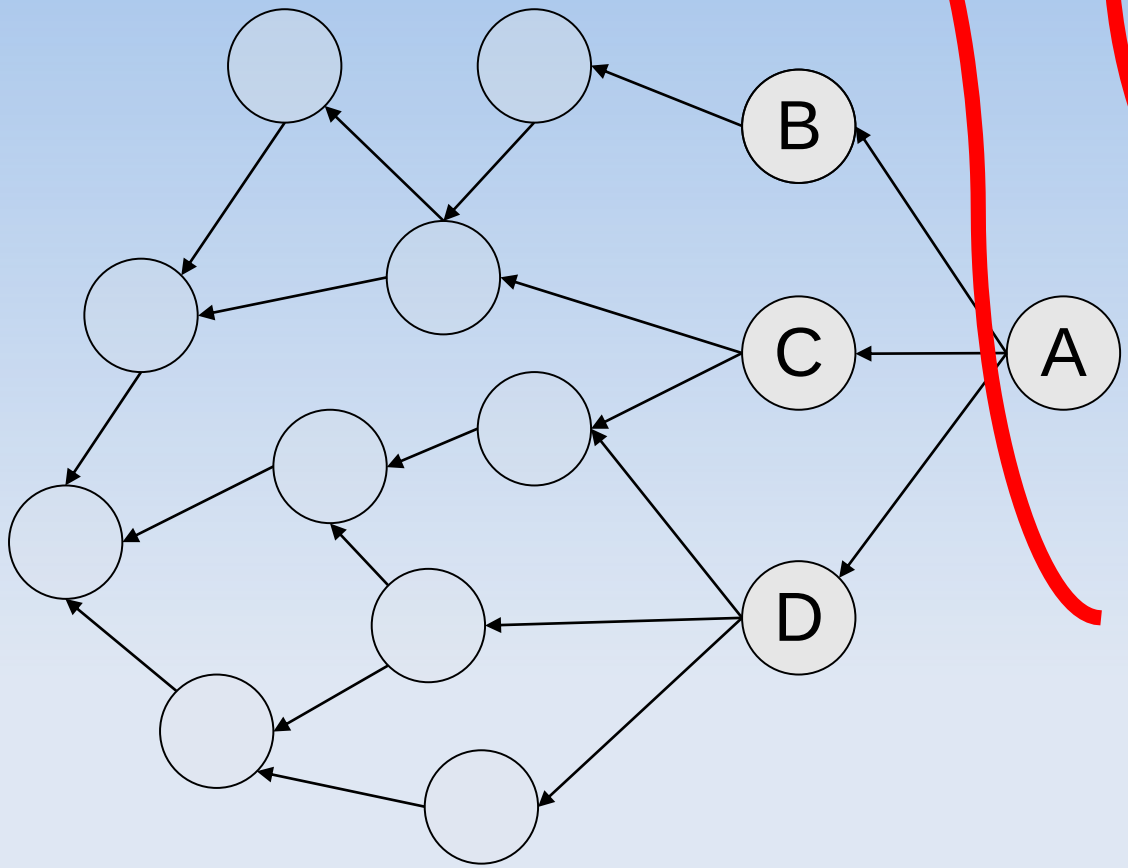
Inefficient API Usage

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

Inefficient API Usage

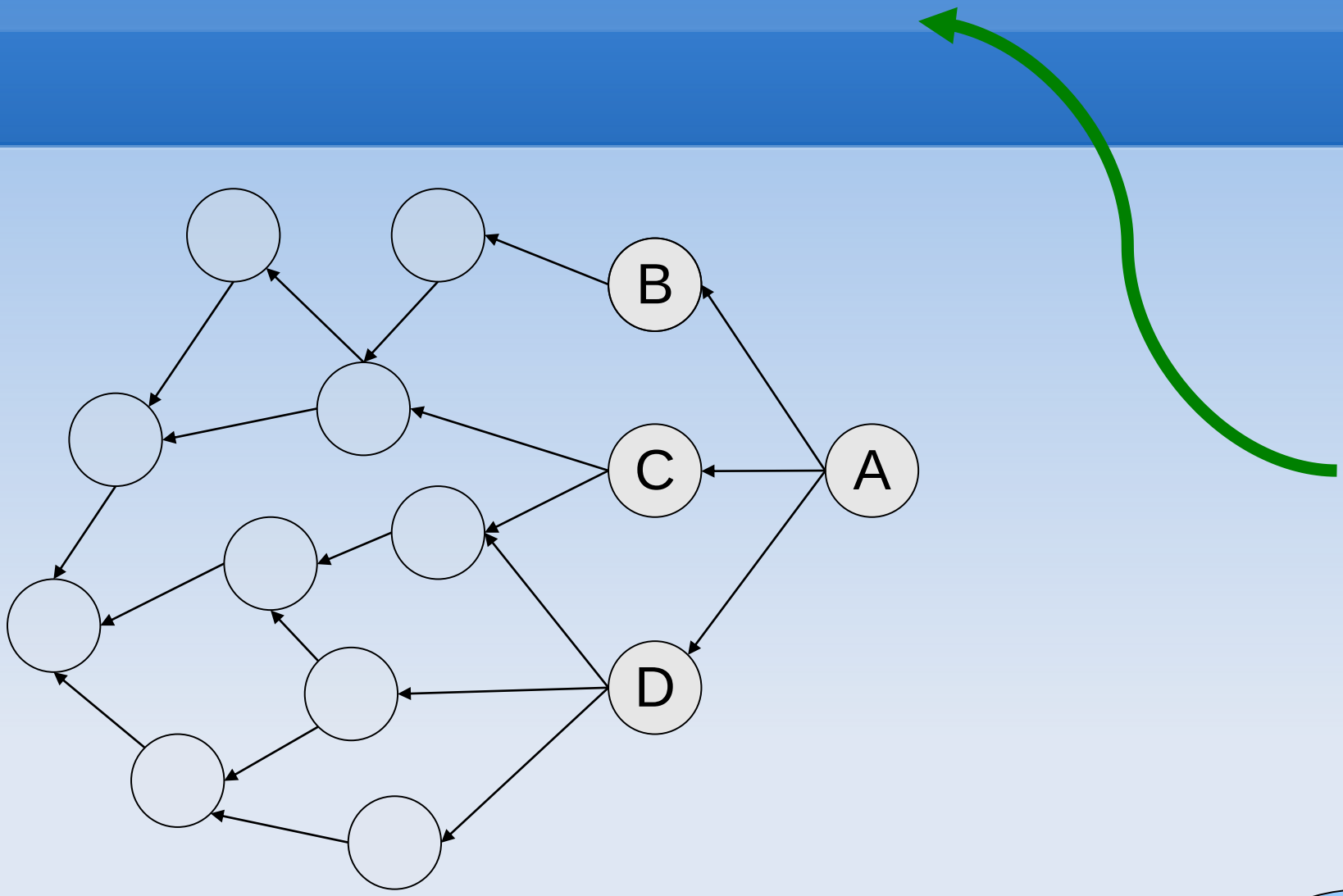
```
gettingURLResponse(String url)
{
    WebResponse response = null;
    URL serverUrl = new URL(url);
    conversation = new WebConversation();
    request = new GetMethodWebRequest(serverUrl,"");
    response = conversation.getResponse(request);
    return response;
}

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

Inefficient API Usage



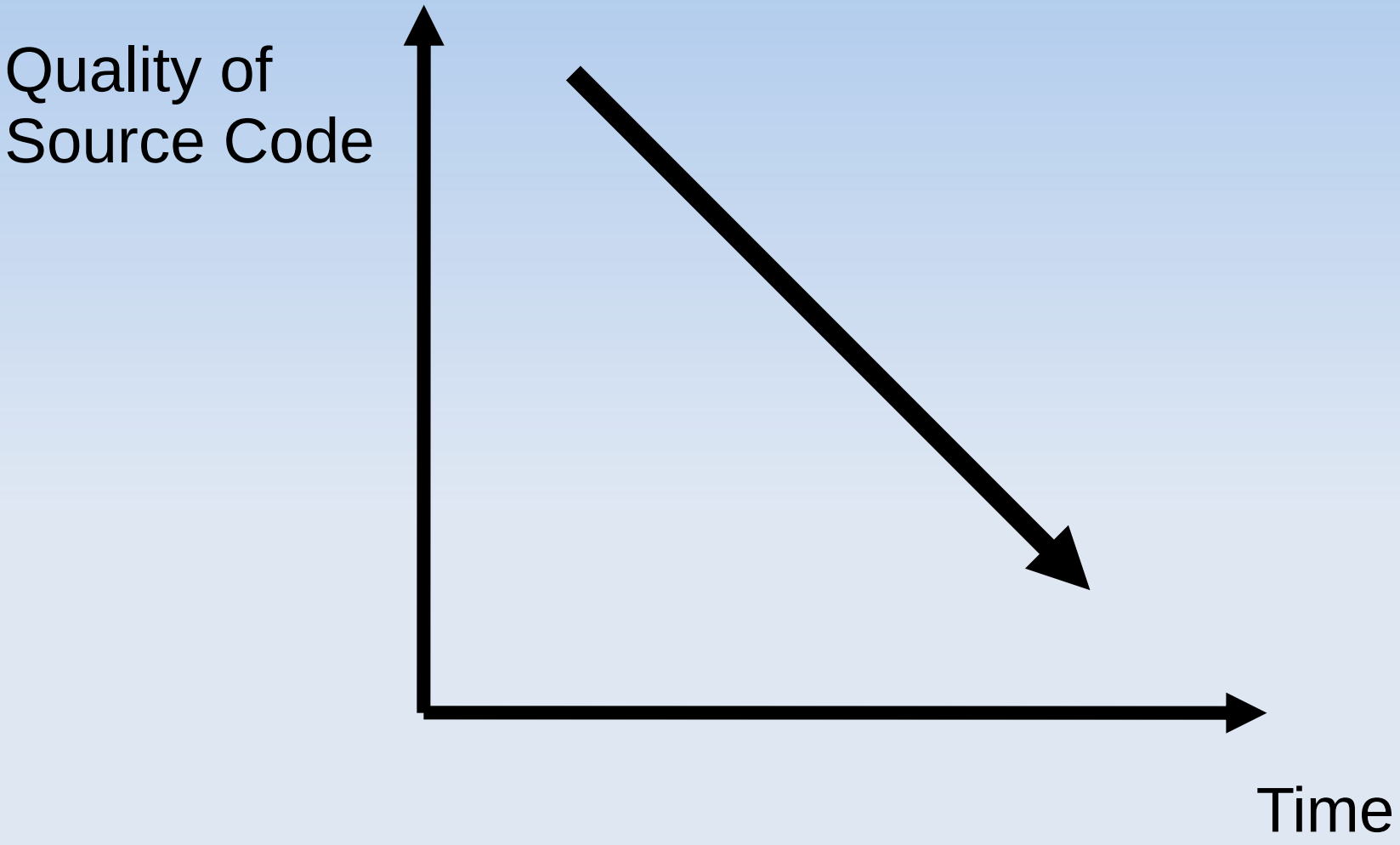
Client Method

Inefficient API Usage



Client Method

Motivation



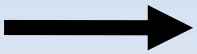
Detection is not Easy

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

getResponse(String url)
{
    return _mainWindow.getResponse(url);
}
```

Our Approach

Abstract Method Bodies

```
gettingURLResponse(String url)
{
    WebResponse response = null;
    URL serverUrl = new URL(url);
    con = new WebConversation();
    req = new GMWR(serverUrl,"");
    response =  con.getResponse(req);
    return response;
}
```



WebResponse

URL

URL(String)

WebConversation()

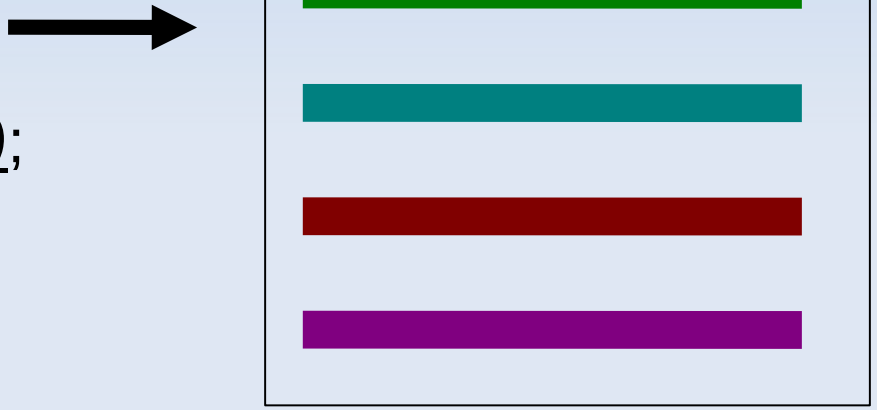
GMWR(URL,String)

getResponse(GMWR)

Our Approach

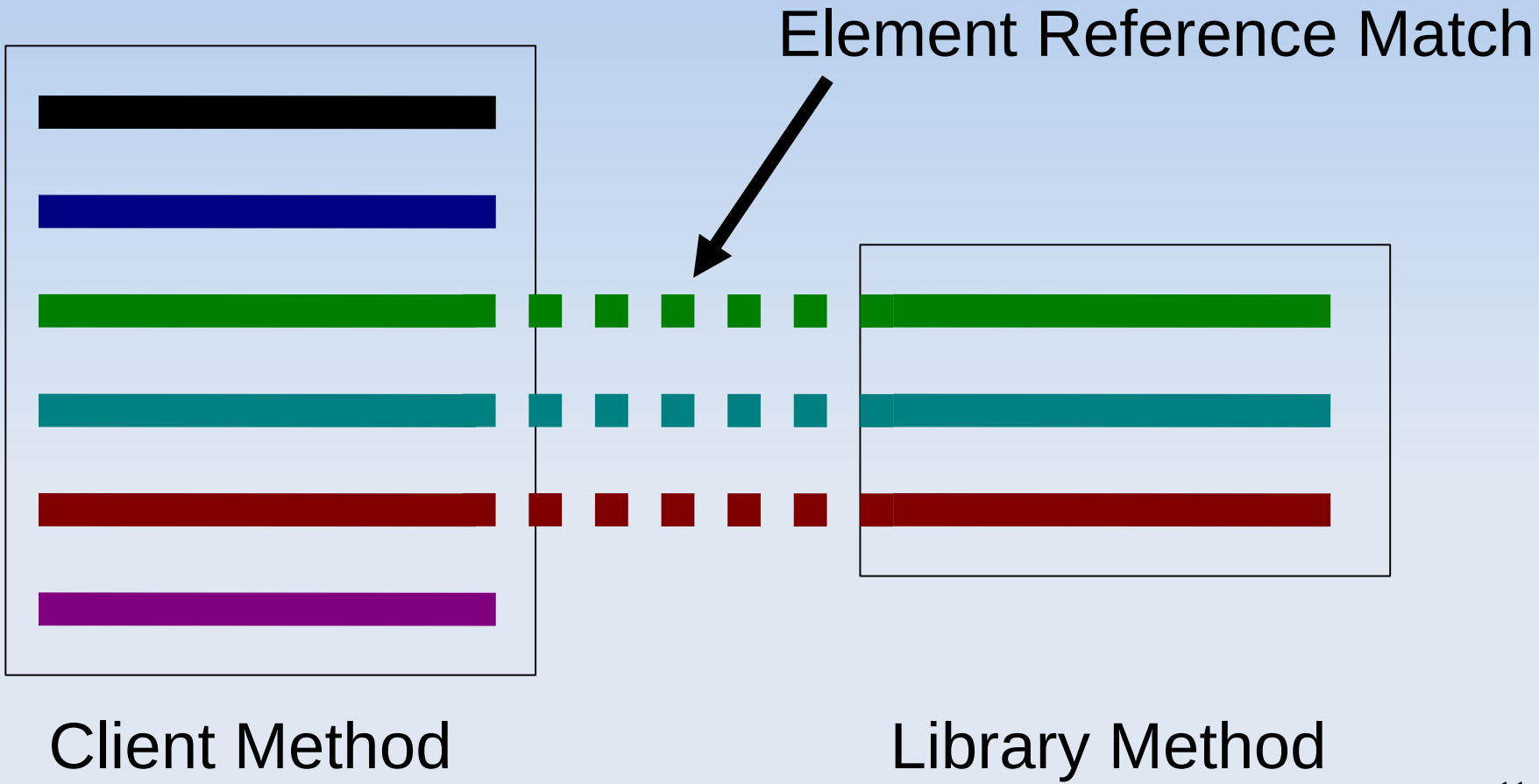
Abstract Method Bodies

```
gettingURLResponse(String url)
{
    WebResponse response = null;
    URL serverUrl = new URL(url);
    con = new WebConversation();
    req = new GMWR(serverUrl,"");
    response =  con.getResponse(req);
    return response;
}
```



Our Approach

Find Imitations

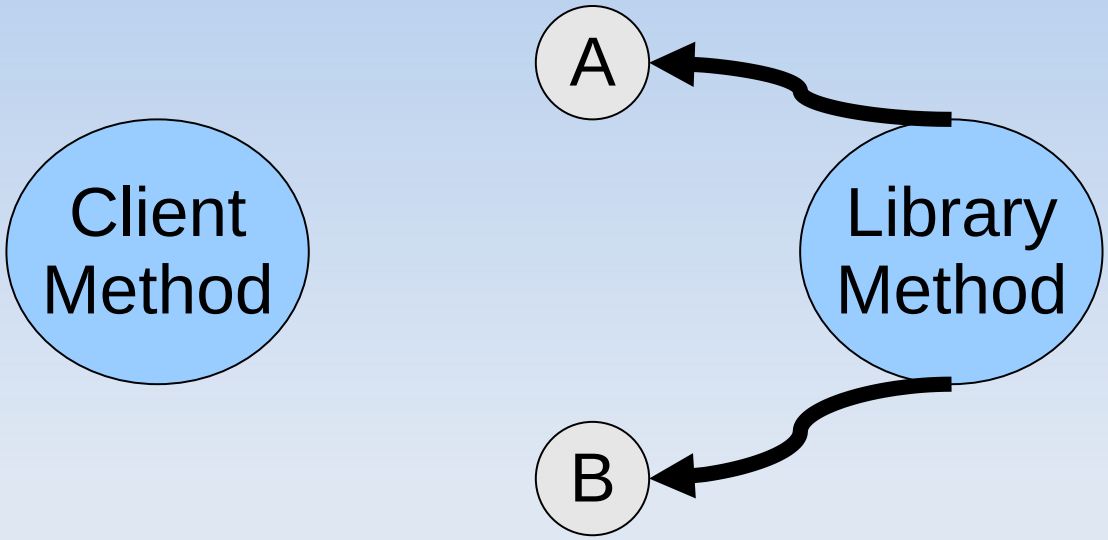


Three Kinds of Matches

Direct

Indirect

Nested

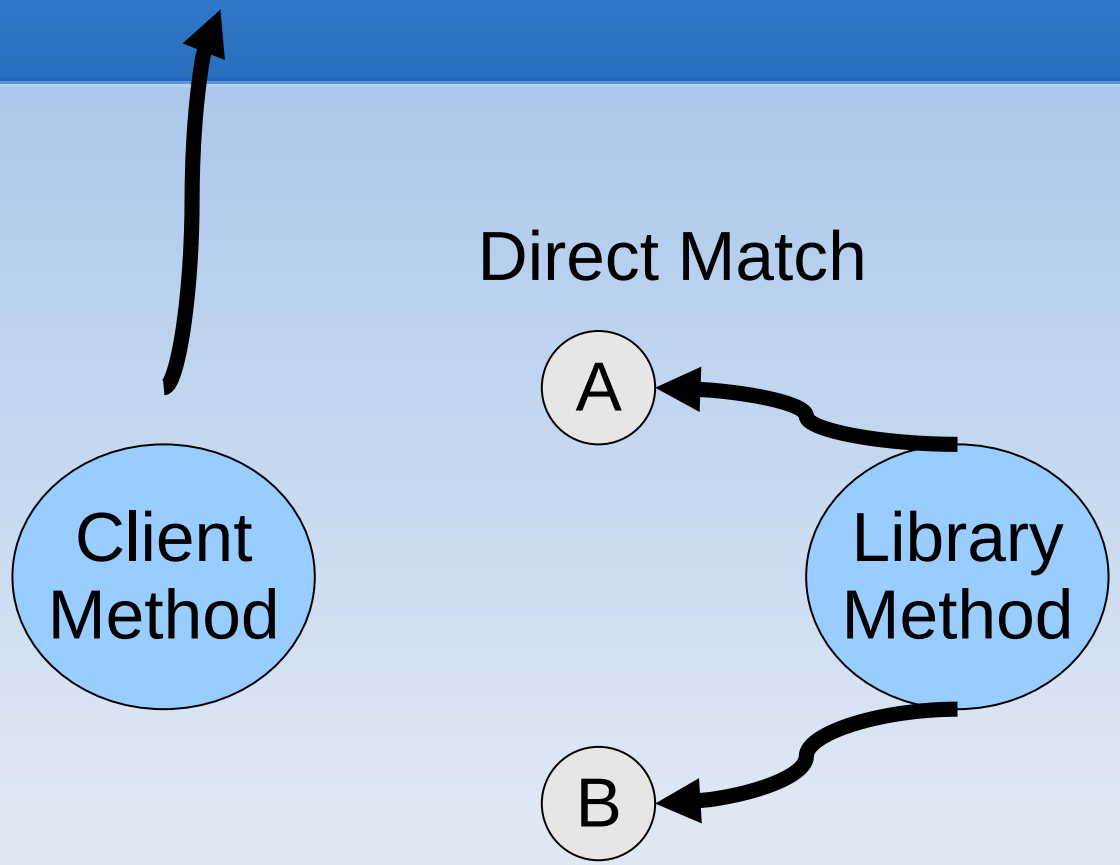


Three Kinds of Matches

Direct

Indirect

Nested

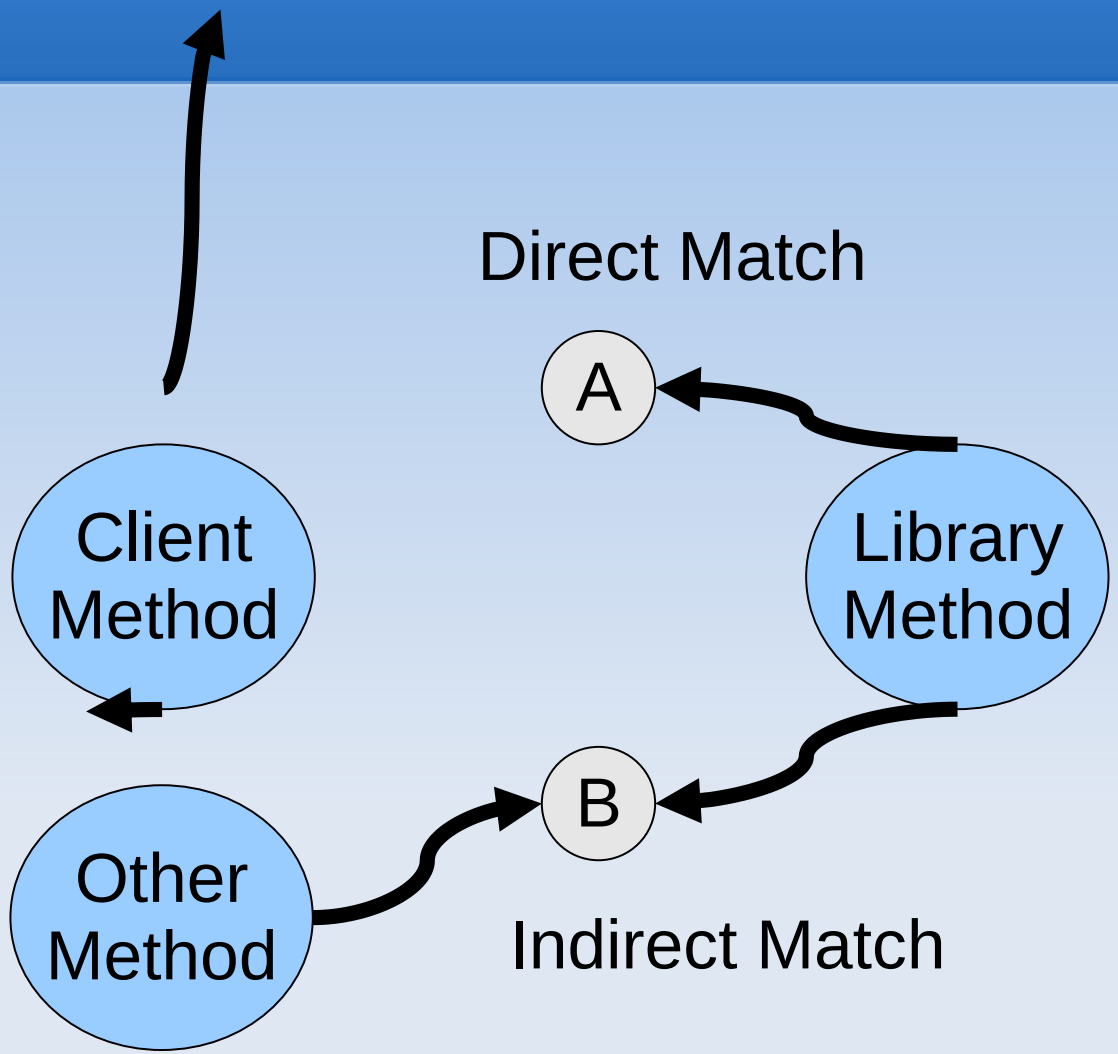


Three Kinds of Matches

Direct

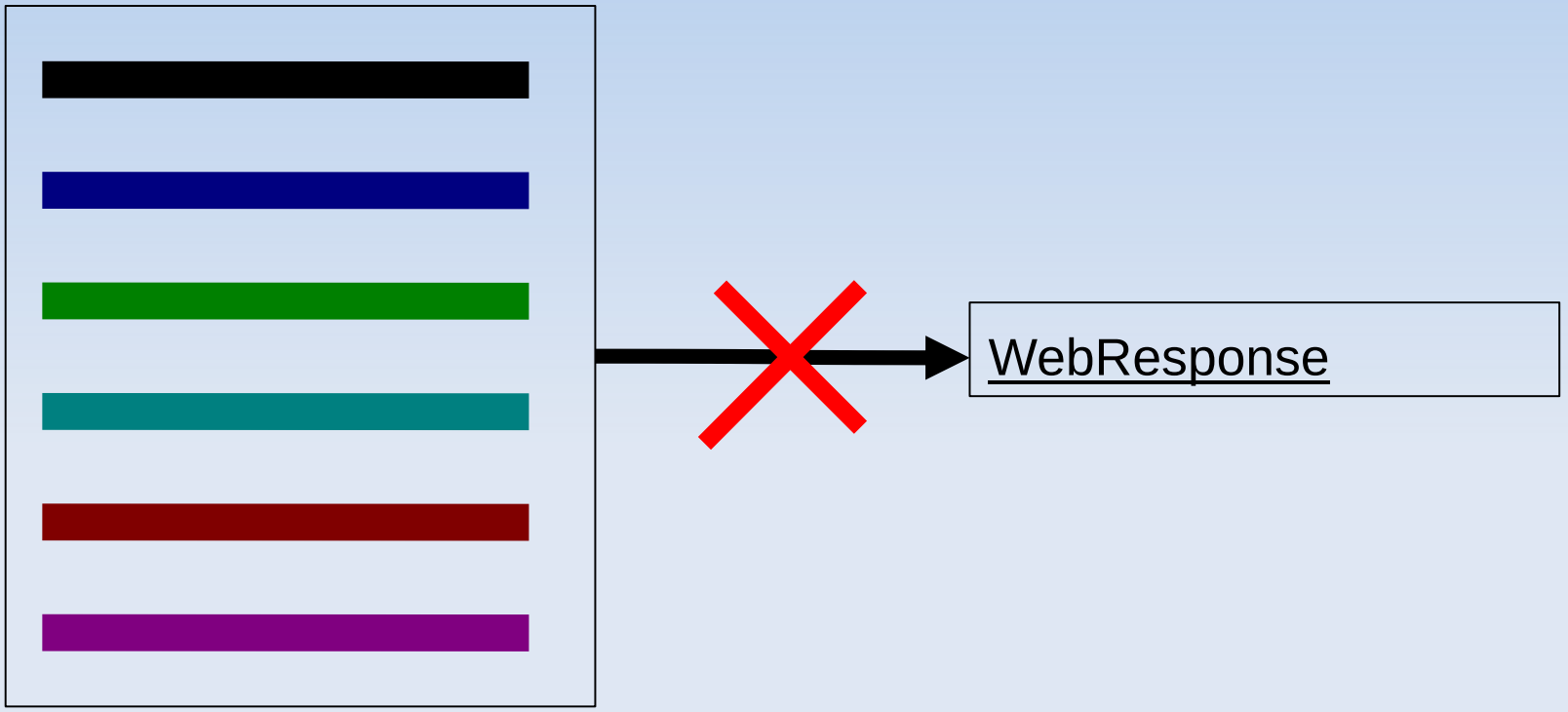
Indirect

Nested



Our Approach

Filter the Imitations

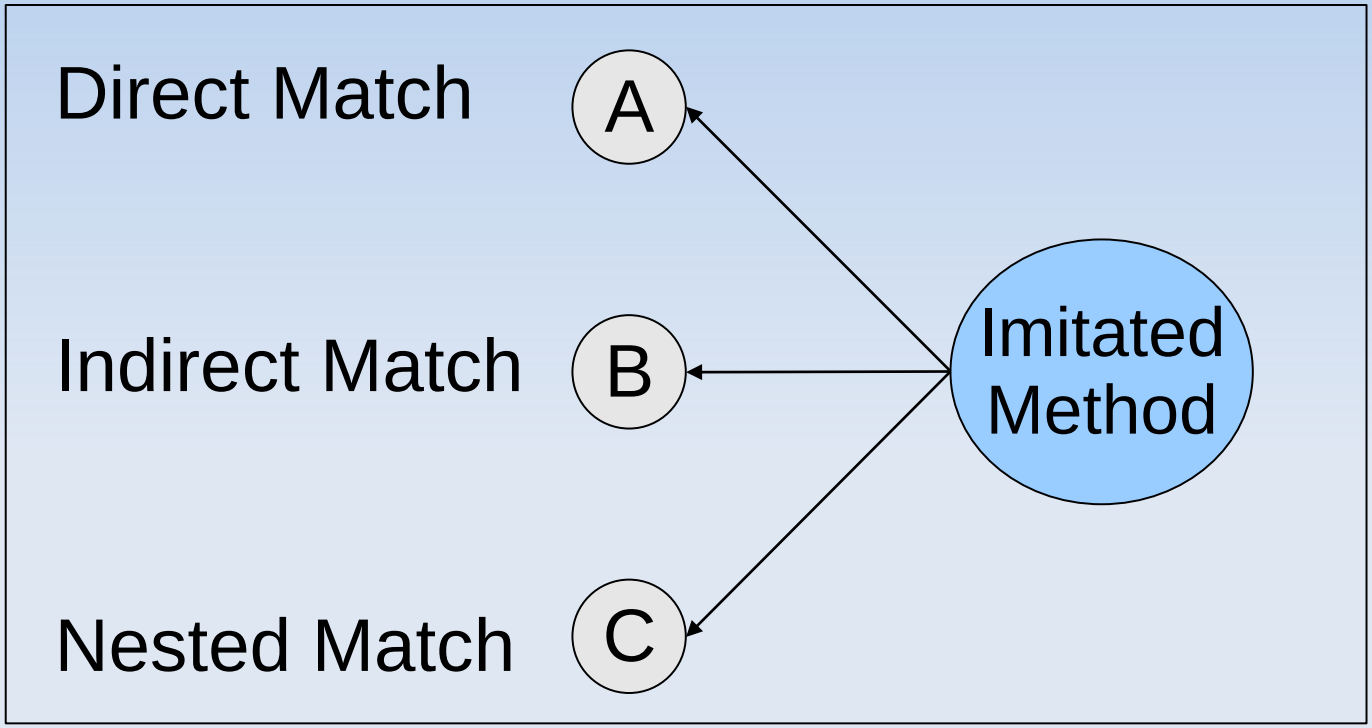


Client Method

Library Method

Our Approach

Abstract Imitations as Usage Patterns

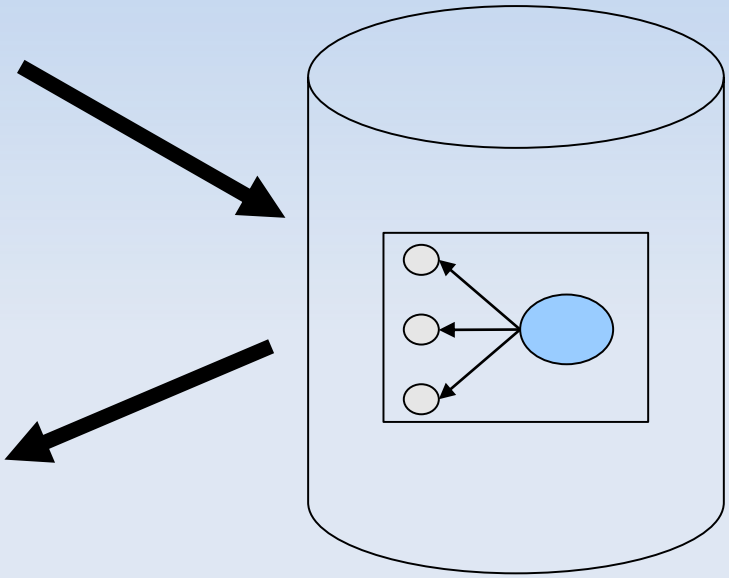


Our Approach

Store and Reuse (good) Patterns

```
gettingURLResponse(String url)
{
    WebResponse resp = null;
    ....
    return resp.getResponse(...);
}
```

```
gettingURLResponse(String url)
{
    WebResponse resp = null;
    ....
    return resp.getResponse(...);
}
```



Preliminary Results

Project	Valid Imitations	Reusable Patterns	"Edits"
JBoss	163	29	209
SpringFramework	1	1	1
JasperServer	13	9	42

Edits: Saving a function call, variable declaration, or String / integer literal.

Typical Case

```
UploadFileSpec[] files = new UploadFileSpec[]
            { new UploadFileSpec(file) };
foo.setParameter(“newData”,files);
```

Vs.

```
foo.setParameter(“newData”,file);
```

Thank You

Questions?