VALIDATING HIDDEN CODE DEPENDENCIES THROUGH RUNTIME TRACES

PRESENTED BY MOHAMED AMINE HADDAJI

SUPERVISED BY HAFEDH MILI

Lightning Talks

June 6, 2019

UQÀM

What are we doing ?

- We are aiming to migrate legacy applications to Service Oriented Architectures.
  - Identify the reusable clusters of features / functionalities that are implemented in the legacy system.
  - A complete dependency graph of an application could be used as input to identify potential reusable services.

- Legacy J2EE applications are :
  - Multi-tiered
  - Multi-languages

Hidden dependencies ?

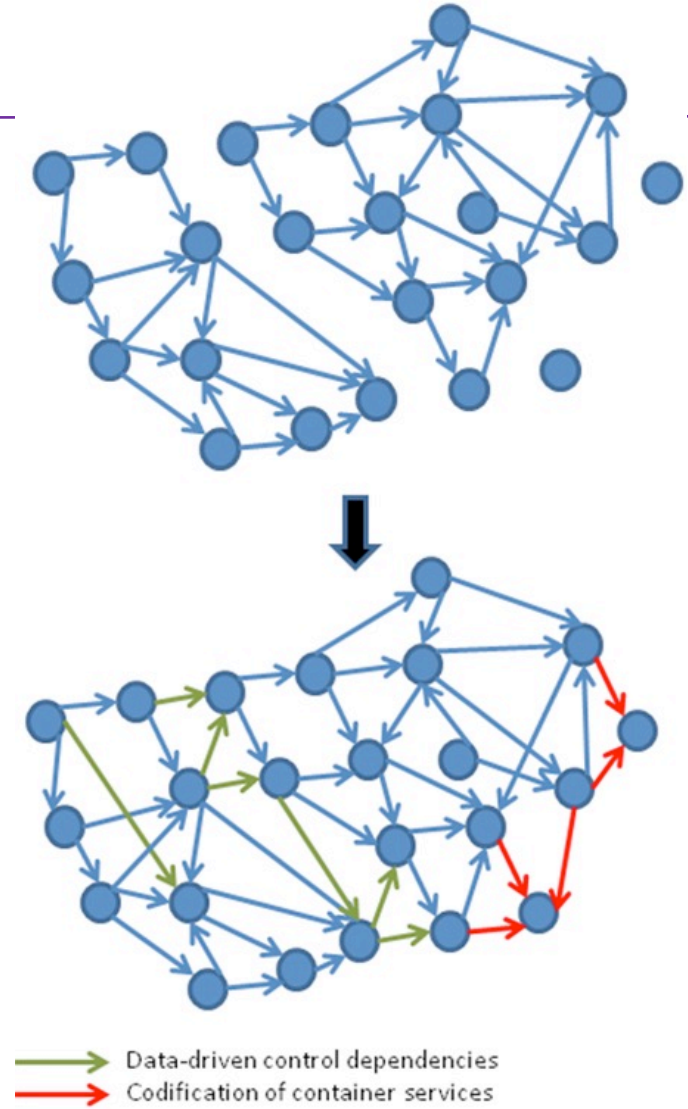
- In order for the client code to use services in the web tier, callback methods and specific interfaces must be implemented.
  - Those are implicit and cannot be seen in the user code.

- Sniff the source code for those calls and artificially add container call dependencies using a rules-based engine.

What are we trying to achieve ?

- Our goal is to validate the added hidden dependencies (implicit calls) by relying on execution traces of those legacy applications.

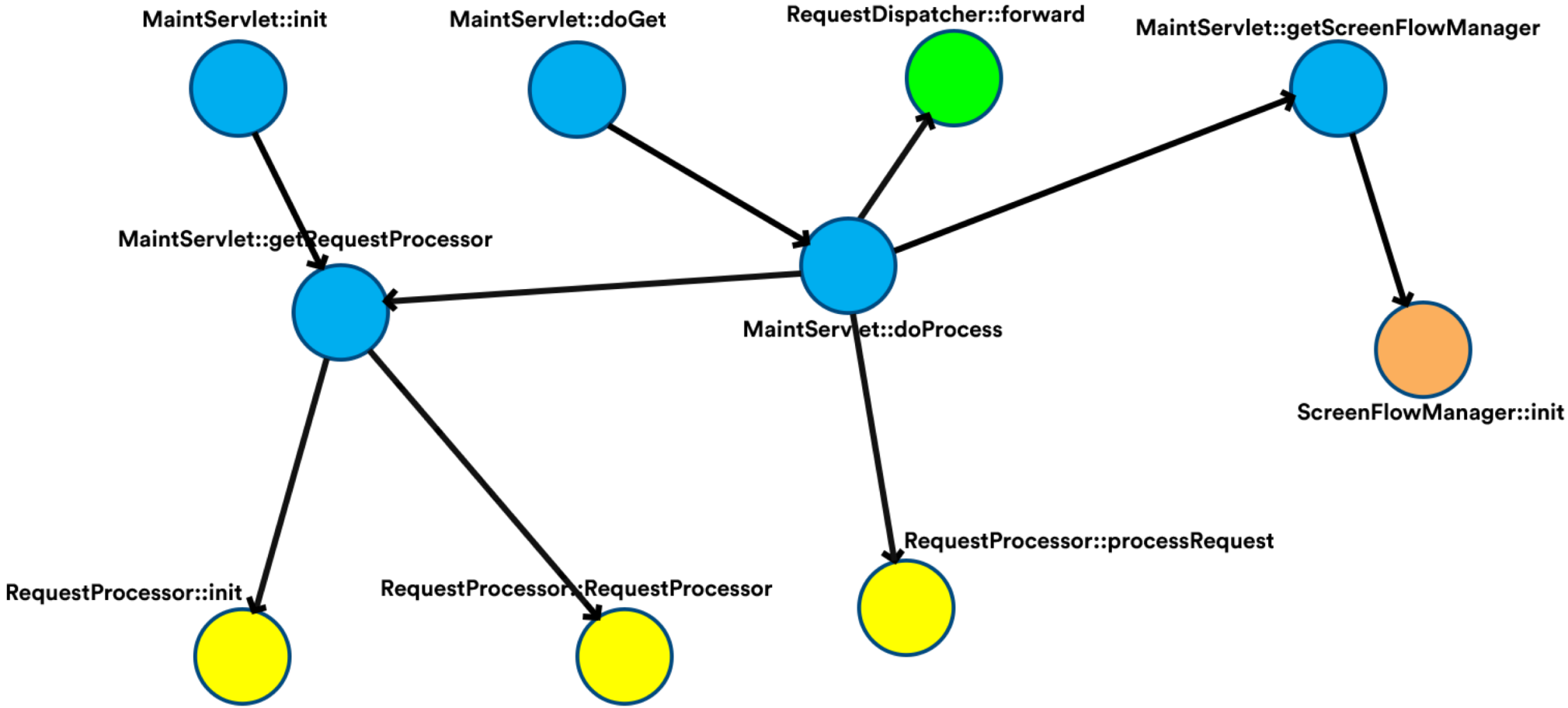
- Dynamic code analysis (run-time tracing)
  - Both client and server must be traced.



IN DEPTH

- Static code analysis
  - KDM™ : a metamodel for knowledge discovery in software.
- Some legacy J2EE applications :
  - Springstore
  - Vaza
  - **Petstore 1.1.2**
- Dynamic code analysis (MaintainJ)
  - Generates runtime sequence diagrams.
  - Traces applications running on a single or multiple JVM's.

Comparing the call graphs



Fully Qualified Names :

com.sun.j2ee.blueprints.control.web.MaintServlet = MainServlet
com.sun.j2ee.blueprints.control.web.RequestProcessor = RequestProcessor
com.sun.j2ee.blueprints.control.web.ScreenFlowManager = ScreenFlowManager
javax.servlet.RequestProcessor = RequestProcessor

What is left to be done ? What we might encounter ?

- Are we going to invoke all the methods ?
- Are we going to find all the call traces in the statically generated call graph ?
- Dynamically, are we going to be able to really detect the calls from the client side straight to the server side ?
  - *Without having the client side pointing to interfaces or proxies