# VALIDATING HIDDEN CODE DEPENDENCIES THROUGH RUNTIME TRACES

PRESENTED BY MOHAMED AMINE HADDAJI

SUPERVISED BY HAFEDH MILI

Lightning Talks

June 6, 2019

UOÀM

# 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.

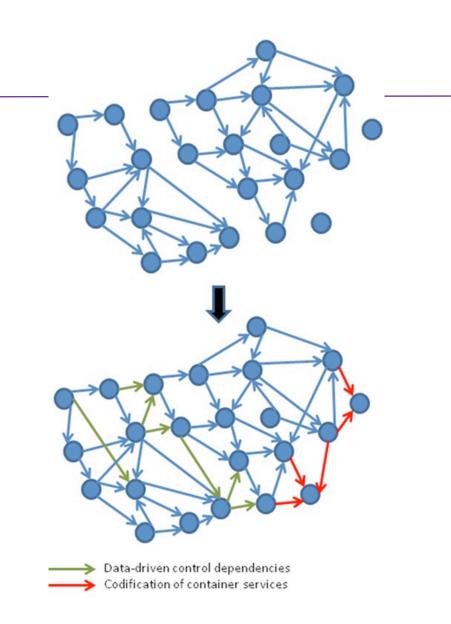
• Sniff the source code for those calls and artificially add container call dependencies using a rules-based engine.

 $\bullet \bullet \bullet \bullet$ 

#### 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**

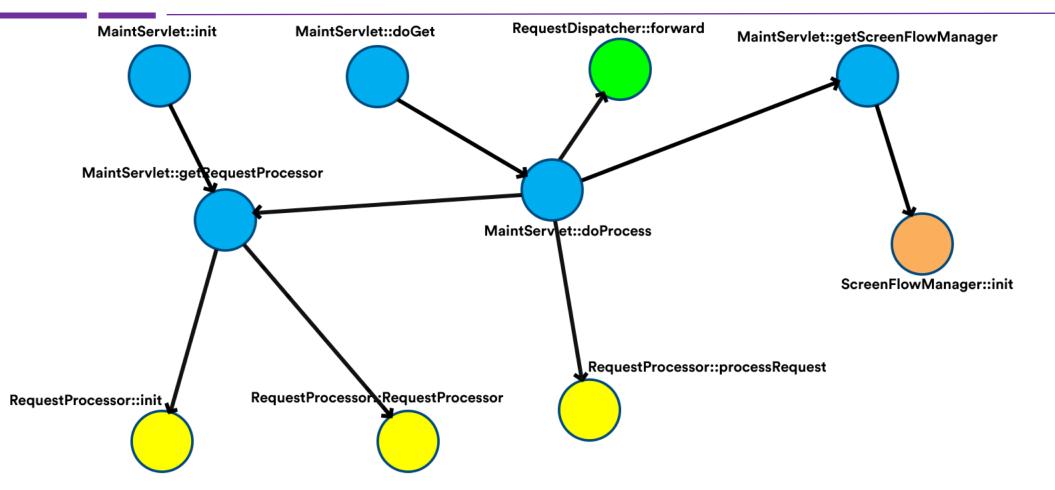
- Static code analysis
  - KDM<sup>™</sup> : a metamodel for knowledge discovery in software.

PopulateServle

- 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.

et	Main	Servlet	Debug	URLMa	pping	Request	rocessor	Languag	eHandler	JSPUti	
						ļ	💻 57:void no	tifyListeners	(Collection a	rg0) (2ms)	
							58:void doEnc	l(HttpServlet	Request and	)) (0ms)	
								(, , , , , , , , , , , , , , , , , , ,		, (	
	ļ	= 59:void	processReque	st(HttpServl	letRequest	arg0) (6ms)					
		₹} 60:Scree	enFlowManage	r getScreen	FlowMana	ger() (0*2 cal	ls = 0ms)				
		<u>.</u>									
	1	= 61:void	getNextScreen	(HttpServle	tRequest a	rg0) (1ms)					_
				-							
				-							
		= 65:void	getNextScreen	(HttpServle	tRequest a	rg0) (1ms)					
	Ì										
	l		doProcess(Htt	pServletRed	quest arg0)	(8ms)					
		67:Locale g	etLocale(HttpS	ession arg	)) (0ms)						
		- 60.Chris		0	0) (0)						
	[	- 08:Strin	g getTemplate	Locale arg	uj (ums)						
		= 70:Strin	a aetTemplate	(Locale ard	0) (0ms)						

#### **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