ServiceCoroner: A Stale References Diagnosis Tool for the OSGi™ Services Platform

Kiev Gama & Didier Donsez

Université Grenoble 1, France

Kiev.Gama@imag.fr

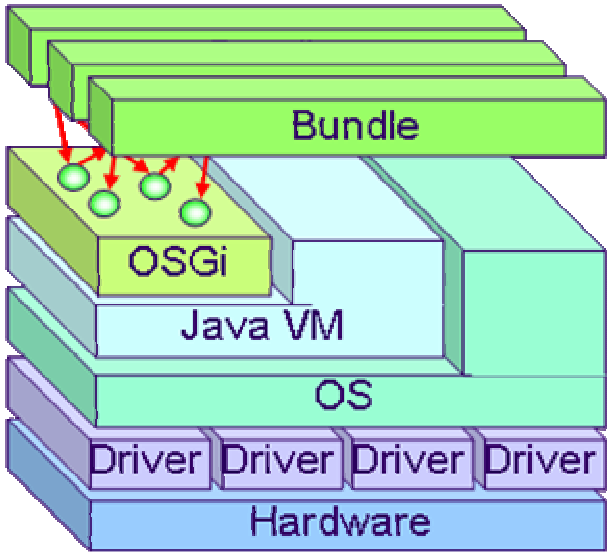
Didier.Donsez@imag.fr

Outline

- What is the OSGi platform ?
- The Stale References Pathology
- Need for Diagnosis
- The ServiceCoroner tool
- Experimentation
- Conclusion
- Perspectives

What is OSGi Alliance ?

- Consortium founded in March 1999
- Objective
  - Create open specifications for delivering administrated Java services through a network
- Define
  - A common platform (framework)
    - Services deployment
    - Services execution and administration
  - A set of based services:
    - Log Service, Http Service…
  - A deployment unit, a ***bundle***

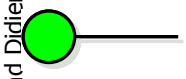


With permission of Peter Kriens

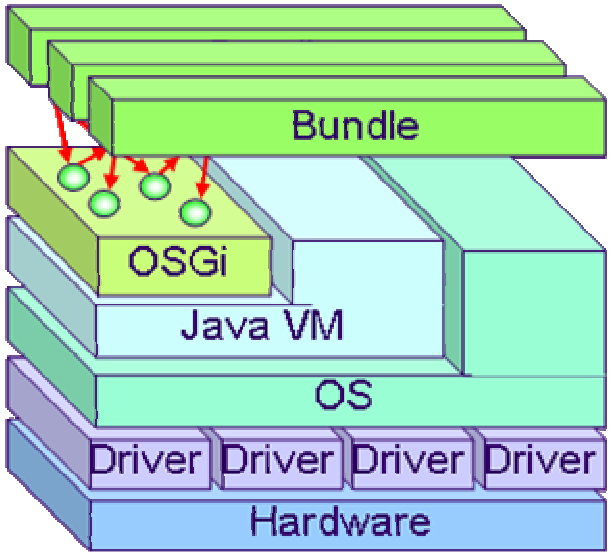
OSGi Main Concepts

- Framework:
  - Bundles execution environment
    - Felix, Knopperfish, Equinox, SMF, ProSyst, …
  - Lifecycle Event notification

- Bundles:
  - Services diffusion and deployment unit

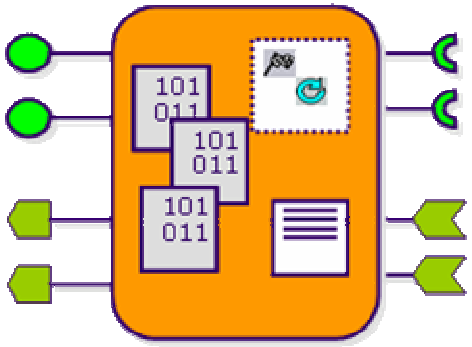


- Services:
  - Java Object implementing a well define contract



OSGi Application Packaging

- Modularize the middleware/application
  - Distribute the different middleware services
  - Better component visibility
  - Need of a deployment container
  - Partial update without restart all



- Implementation
  - Based on Jarfile and Manifest entries
  - Explicit Package dependencies and Versioning (range, …)

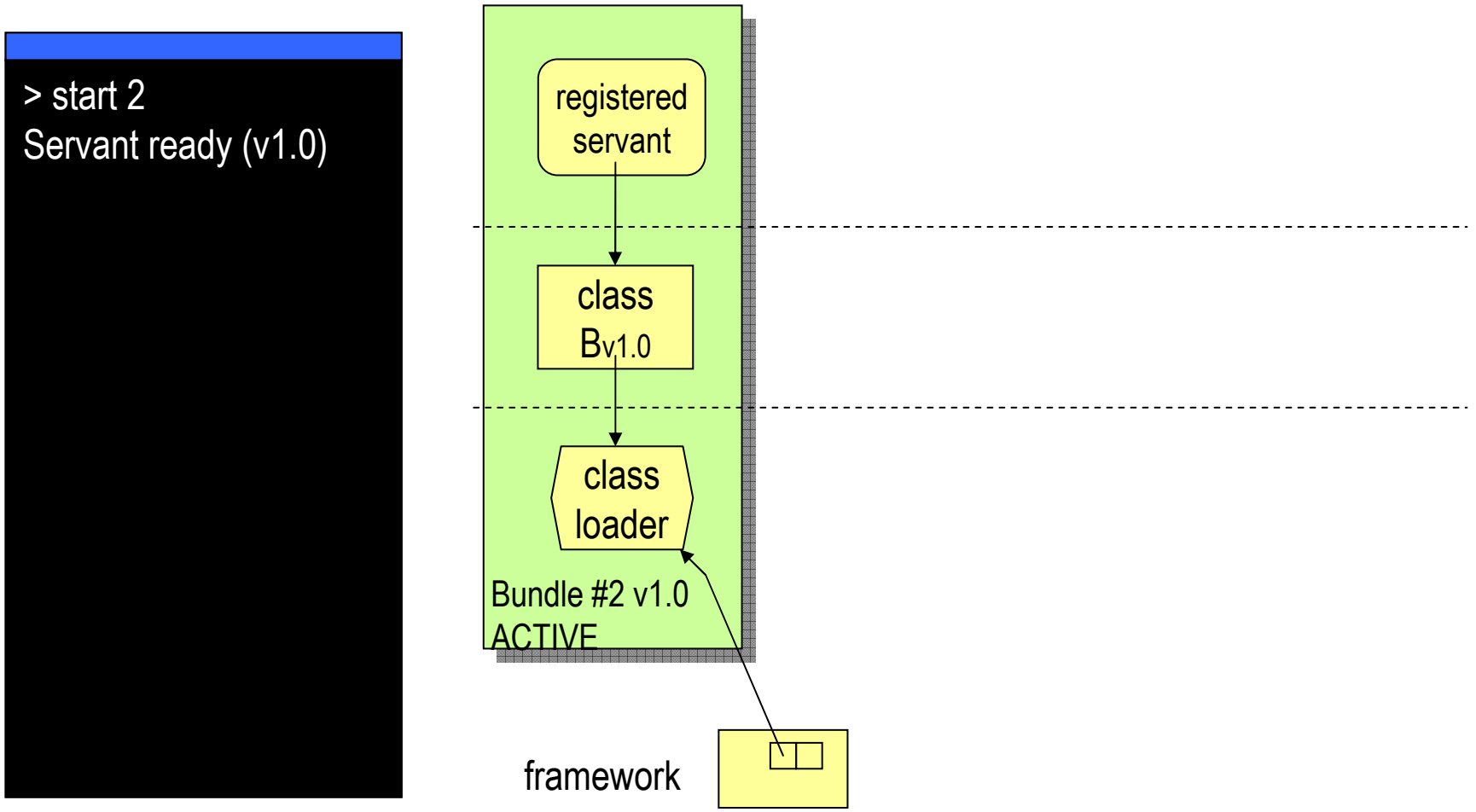
- Ready for probably next generation standard
  - Overtake JNLP (JSR-56), J2EE EAR, OSGi R3 bundle
  - JSR 277 (Java Module System) for Java Platform 7.0

What are Stale References?

"a reference to a Java object that belongs to the class loader of a bundle that is stopped or is associated with a service object that is unregistered"

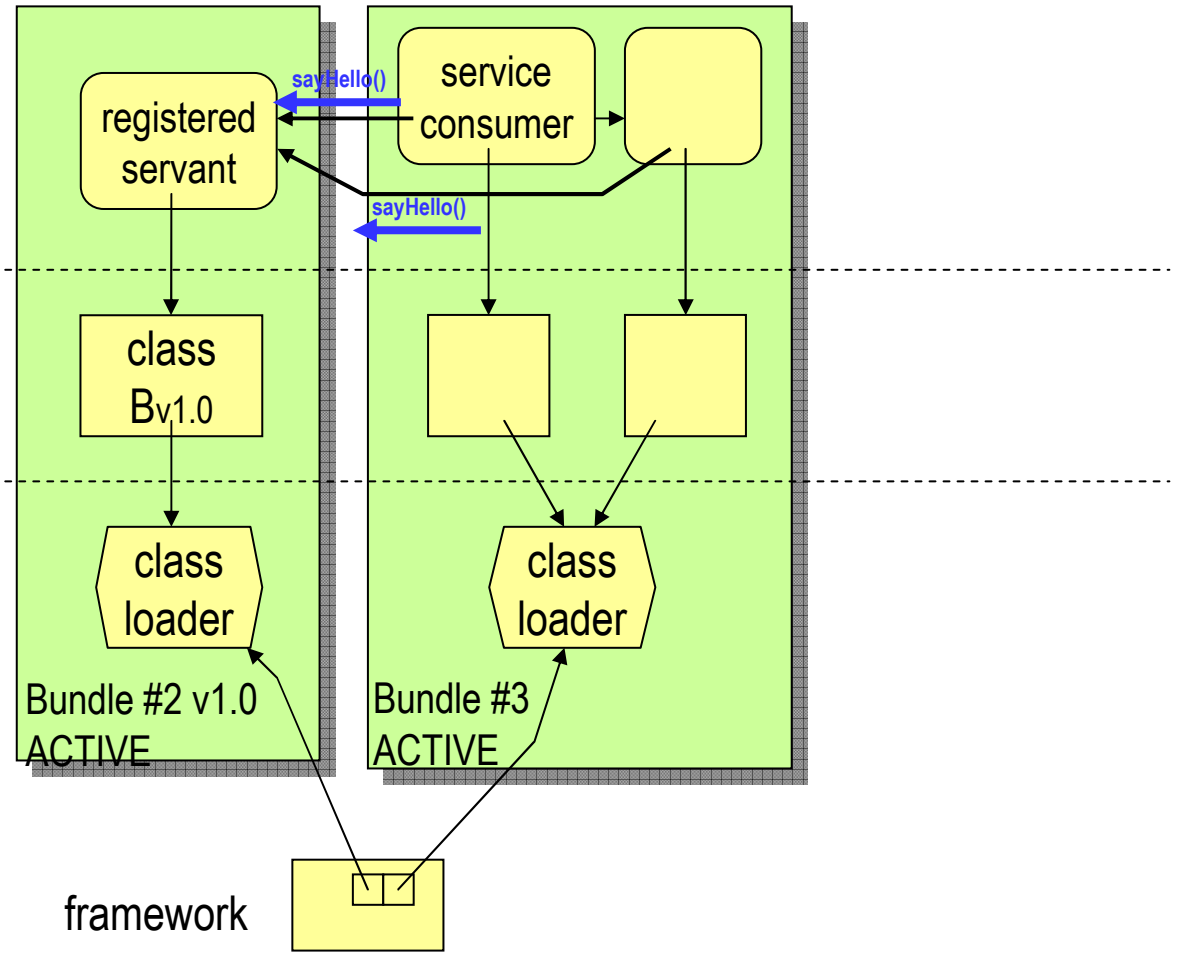
OSGi R4 Section 5.4

An example of Stale Reference Pathology? (i) initial



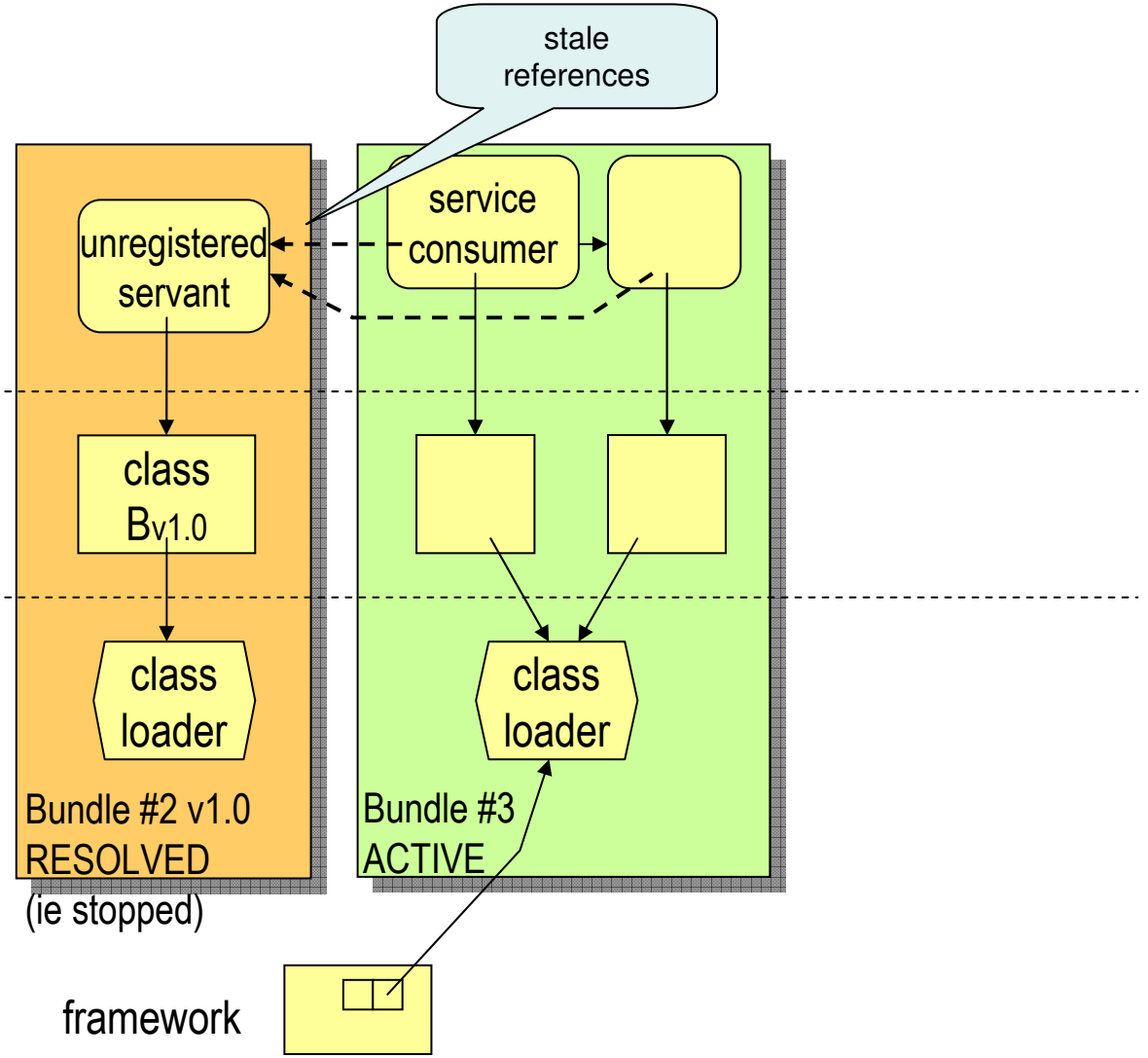
An example of Stale Reference Pathology? (i) initial

```
> start 2
Servant ready (v1.0)
> start 3
1- Hello World ! (v1.0)
2- Hello World ! (v1.0)
```

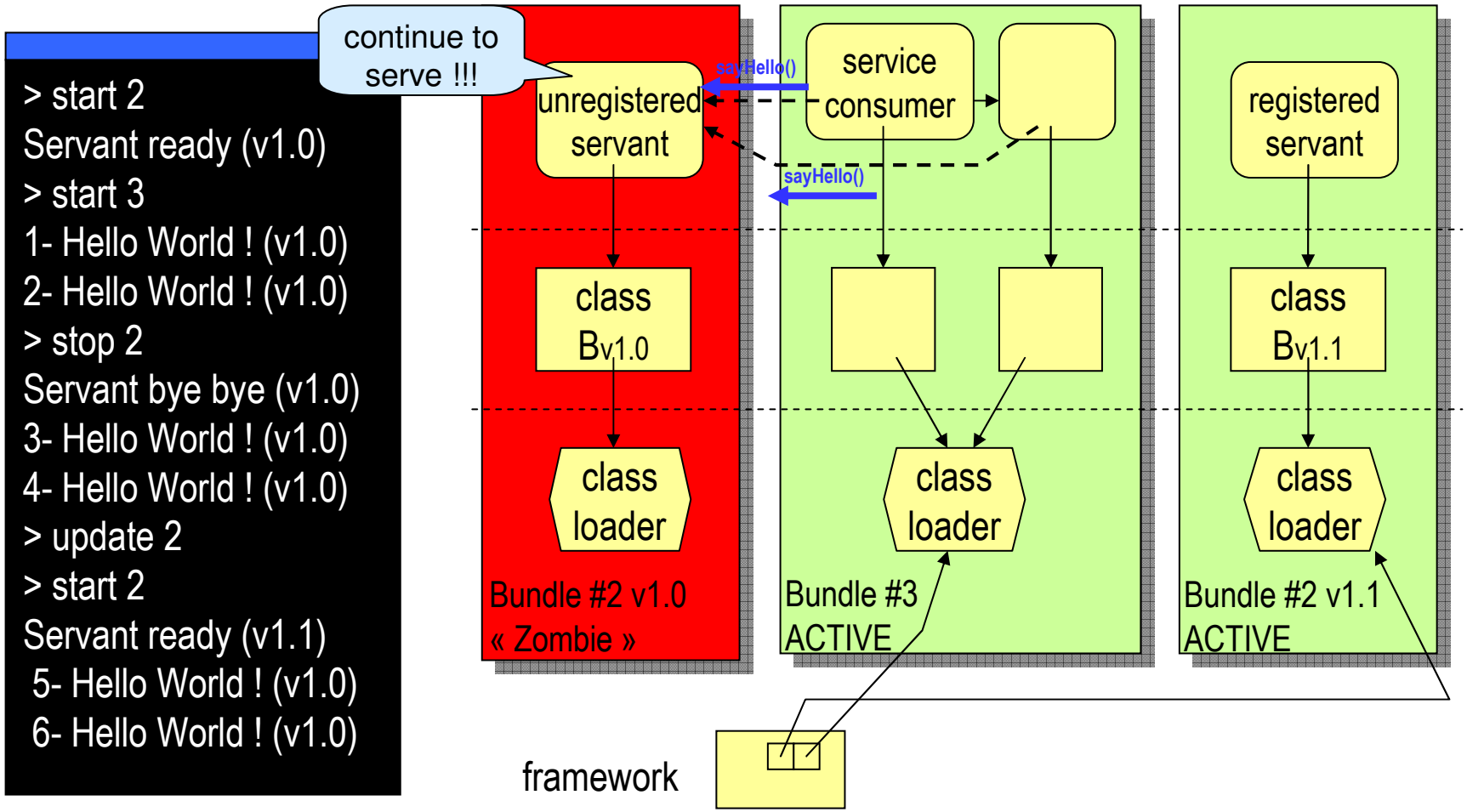


An example of Stale Reference Pathology? (ii) After stop 2

```
> start 2
Servant ready (v1.0)
> start 3
1- Hello World ! (v1.0)
2- Hello World ! (v1.0)
> stop 2
Servant bye bye (v1.0)
```



An example of Stale Reference Pathology?
(iii) After update 2 & start 2



Bad Consequences

- Memory leaks
  - Retention of the classloader of a stopped or uninstalled bundle
  - Retention of all java.lang.Class loaded by that bundle
- Utilization of invalid services → Inconsistencies!
  - Service is unregistered but still used (wrong!)
  - Its context is most likely inconsistent
    - e.g. closed connections, old date
  - Possible exceptions upon service calls
    - good because we can see the problem
  - Silent propagation of incorrect results (worst case!)
    - E.g. Returning old cached-data

Other « stale » pathologies

- “Forwarded references”
  - From one bundle to another
- “Stale” threads
  - bundle has stopped but created threads has not
- Unregistered MBeans, RemoteObjects, …
- Unreleased resources
  - sockets, file descriptors, locks, …

How to ensure « stale reference free » applications?

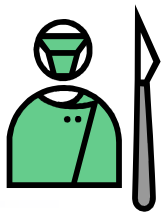
- 2 cases of OSGi™ applications
- From-scratch OSGi™ development
- Bundlization of Legacy codes
  - Really frequent
  - Module with or without Services/Extension Points

- Good OSGi™ programming practices
  - Who trusts their developers ?
- Component Models
  - Necessary but not enough

- Stale references may be there but we can't see them…

- → We need Diagnosis
  victim bundles x guilty bundles

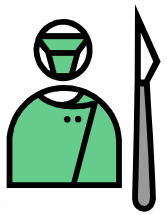
The ServiceCoroner tool



- A diagnostics tool for detecting stale references in OSGi™ applications
- “Inspector” of services death
- Runtime diagnosis
- Points out victim bundles/services and possible suspects

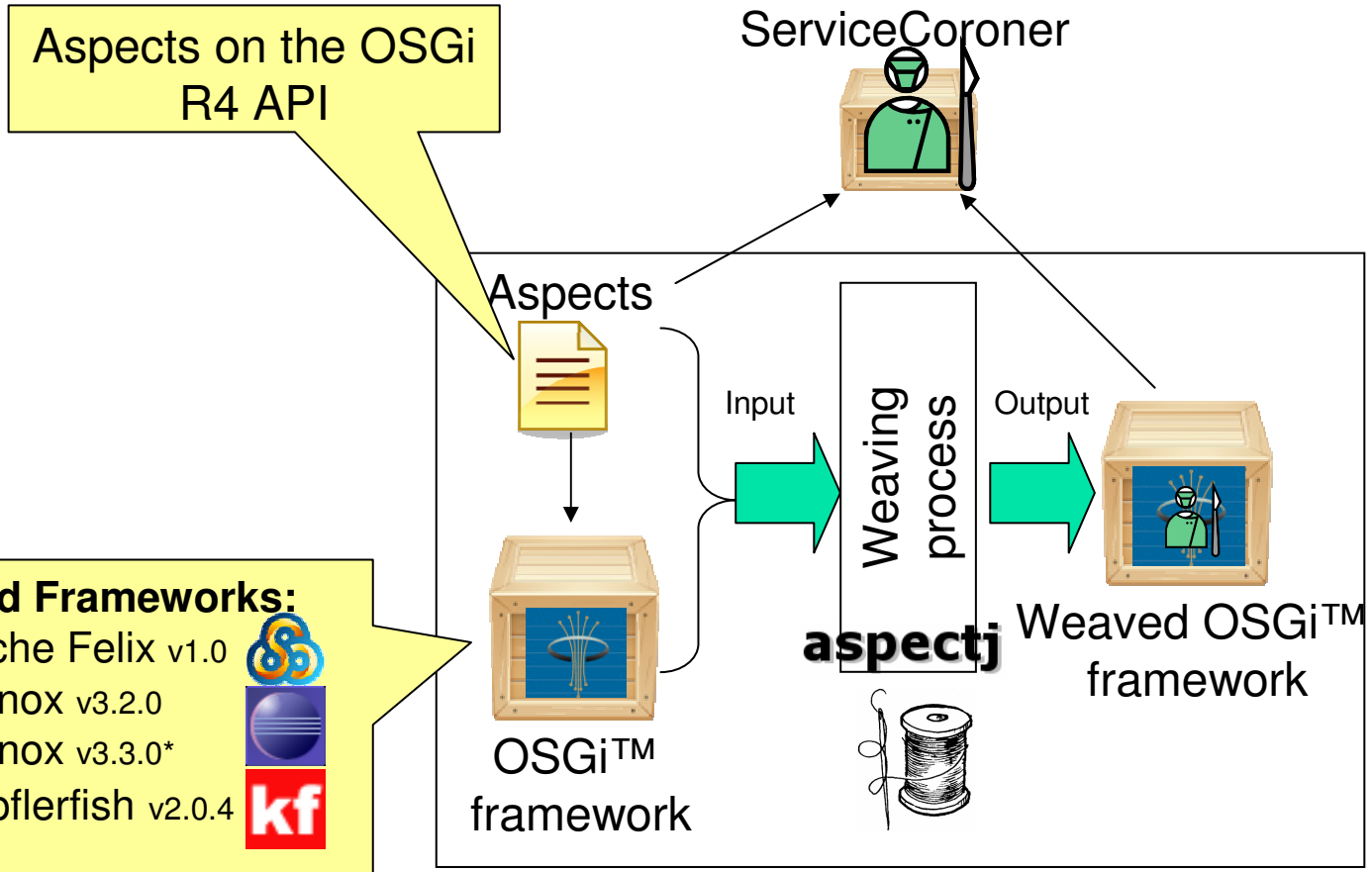
*The coroner is a legal examiner that investigates the causes of unnatural deaths in English speaking countries. Not all coroners have forensic pathology knowledge but for illustration purposes we have named our tool as ServiceCoroner.

The ServiceCoroner tool (cont.)



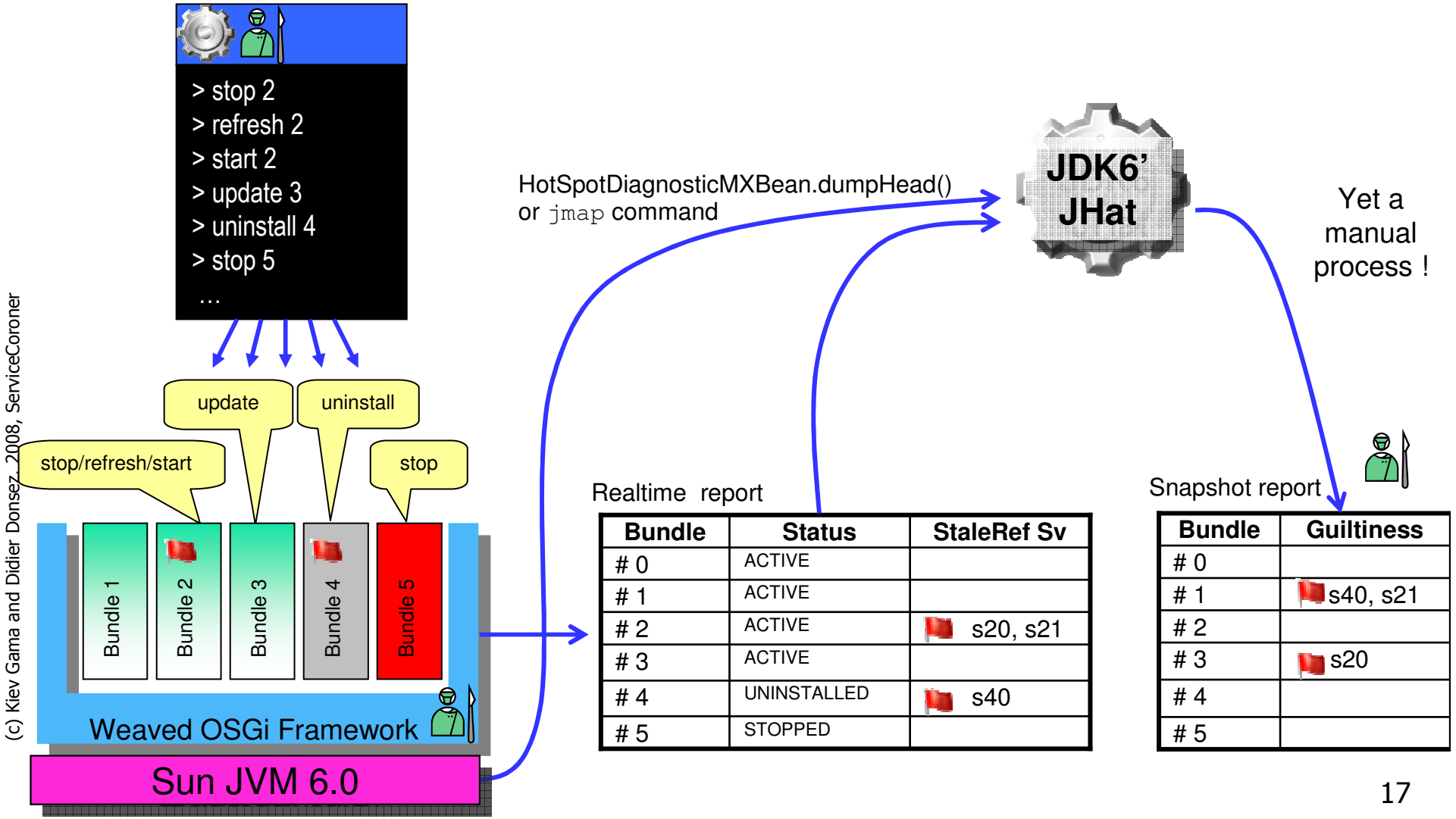
- Diagnosis of service references “pathologies”
- How to enable OSGi™ to provide that info?
  - Use AOP: diagnosis as a separate concern; portability
- Relies on weak references to know if a service has been GCd
  - Small delays (wait for GC) to get actual info
- Listens to service and bundle events and log them
- Minimal performance impacts
  - Weaving Service Registration; Class Loader and Thread Creation

The Weaving Process

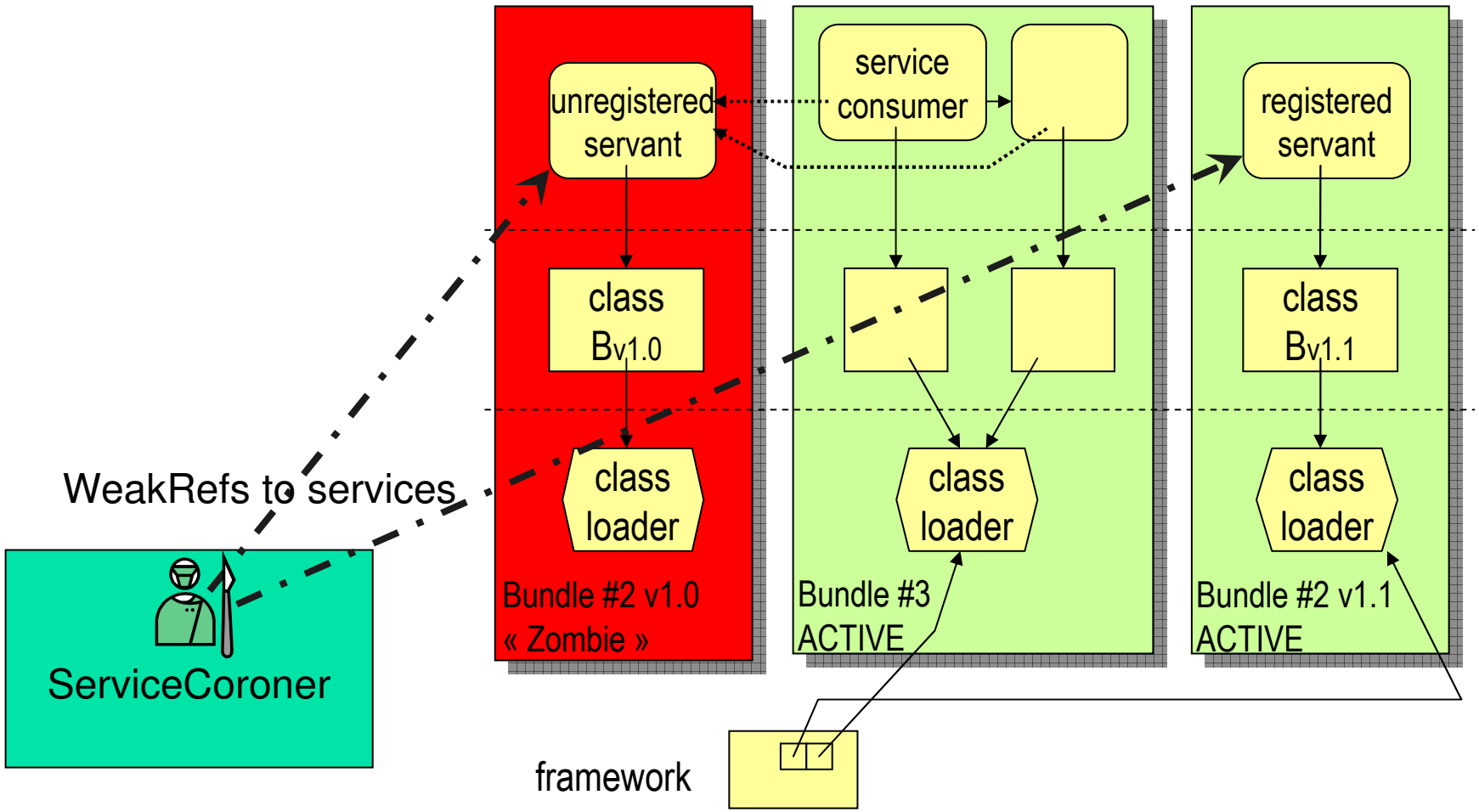


* That version uses signed jars. We manually removed the class hashes information from the original equinox jar manifest in order to bypass checking

The Diagnosis Process



Watching services

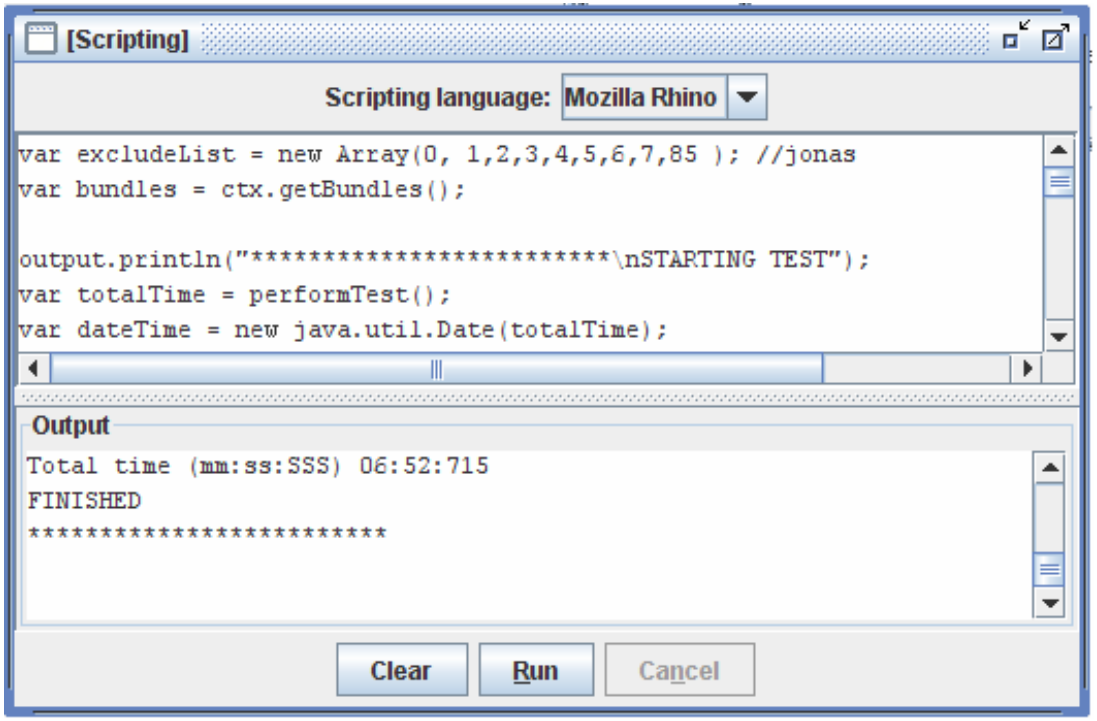


The Diagnosis Process (cont.)

- In vitro (active)
  - Force life cycle events
  - Not ideal for a production environment.
  - Reasonable for a testing environment
  - Faster results
  - "Brute force" may not lead to events that reflect the application's architecture
- In vivo (passive)
  - Wait for "normal" life cycle events
    - resulted from normal administration tasks
  - Ideal for production environments
  - Results are more precise
  - Take longer (maybe days!)

Executing the Active Process Diagnosis

- Run a script in the ServiceCoroner scripting console
- Script performs a call to update in bundles that have registered services
- 10 second interval between each update call
- Core bundles are not updated (e.g. bundle 0, libraries, …)
- Use an “exclude list” containing such bundles

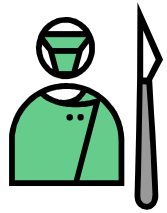


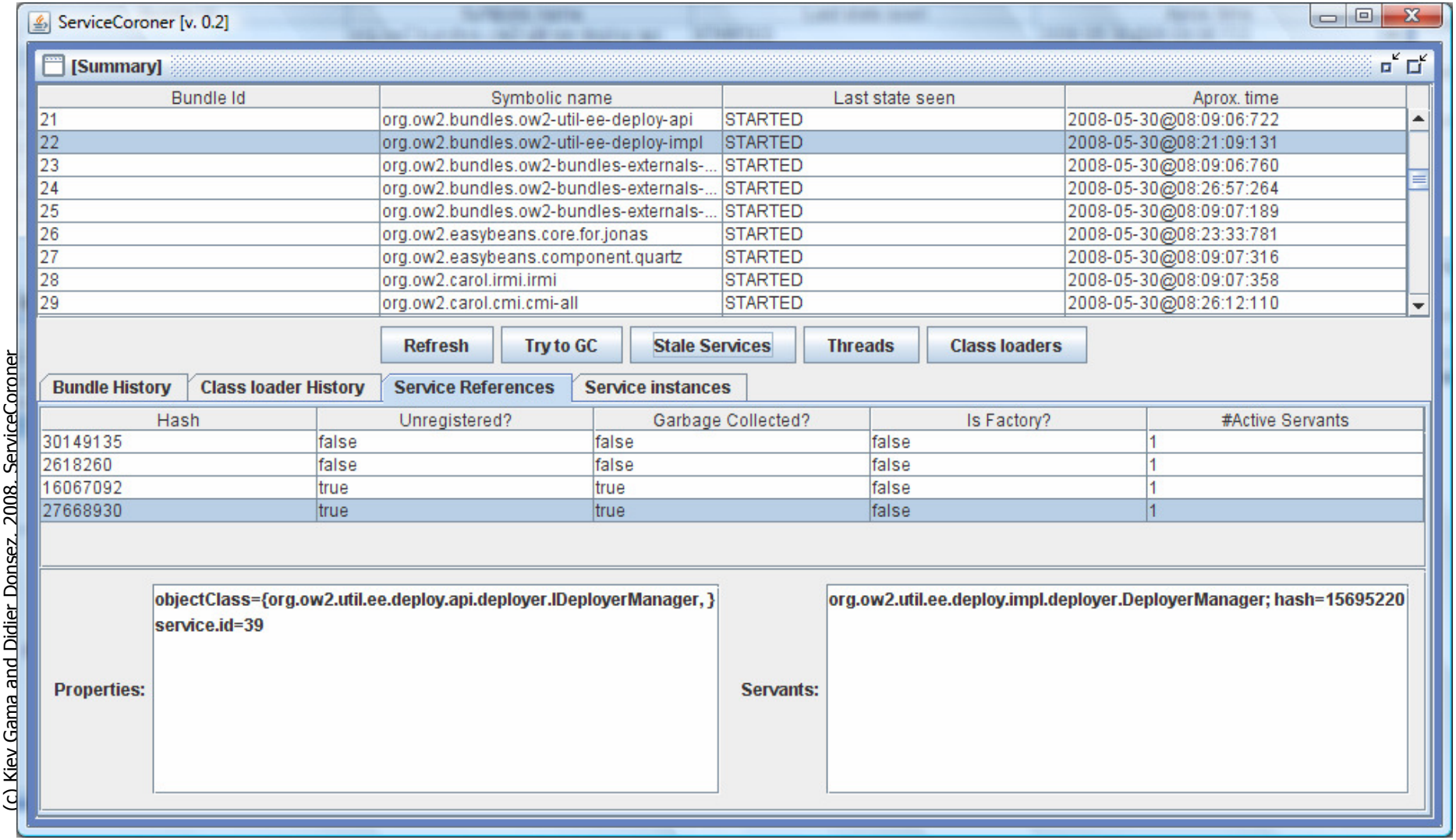
Issues

- Fine grained analysis to find out object referrers
  - Used jhat and jmap embedded in the application
  - Semi-automated process
  - Only in Sun JVM
  - Limitations: Large memory footprint;
  - Weaving at bundle load time

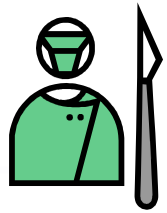
- How to find out the bundle classloader
  - During bundle activation is fine, but…
  - …what about the extender model case and library bundles?
  - We need an accurate mechanism to infer a bundle's classloader

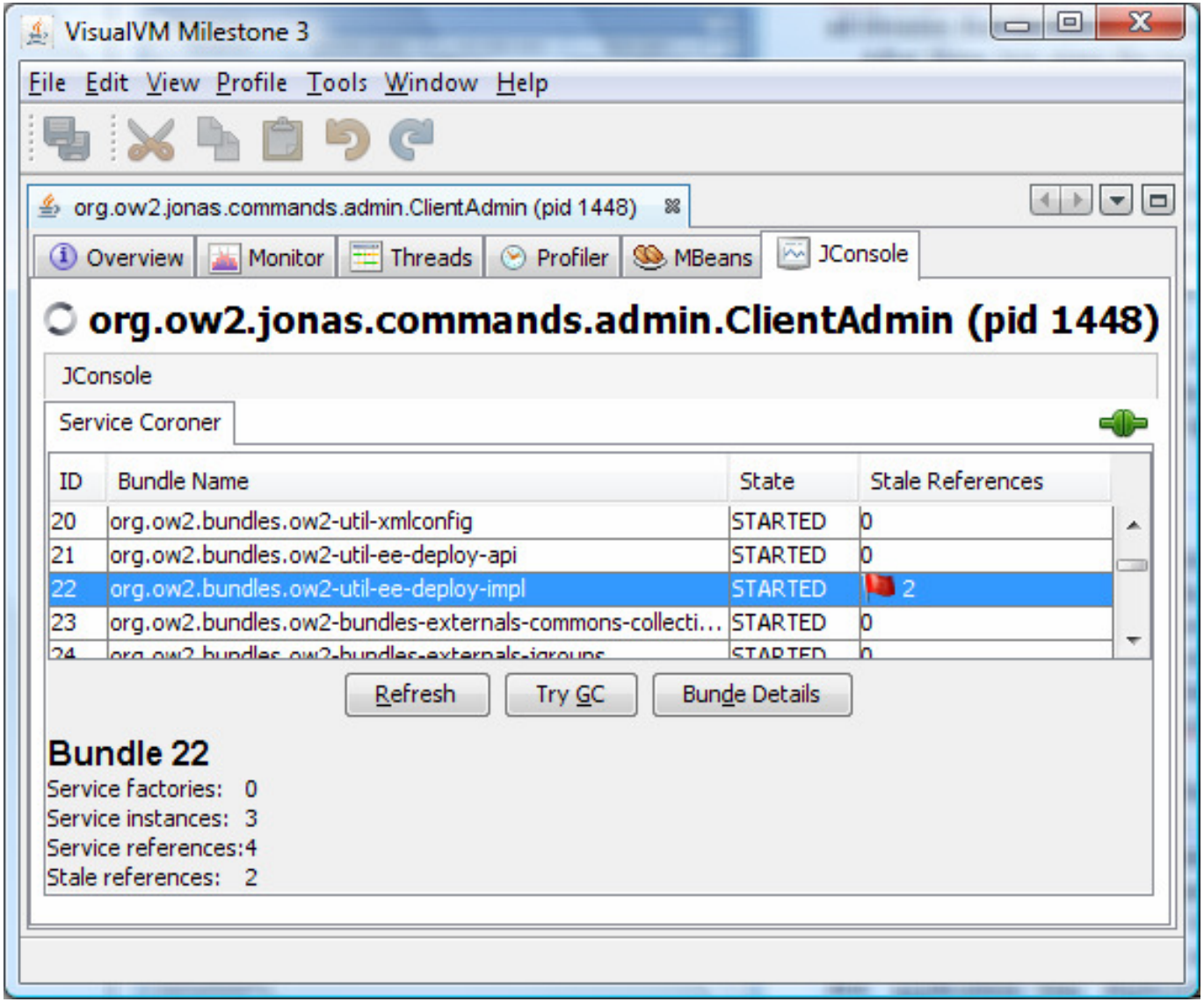
ServiceCoroner Graphical User Tools (i) Standalone





ServiceCoroner Graphical User Tools (ii) JConsole/VisualVM Plugin





Experiments with ServiceCoroner

- Motivation
  - Validate ServiceCoroner on real-life OSGi-based SW
    - Widely used
    - OSS and Non-Commercial OSGi apps
      to avoid court trials or man hunts ;-(
    - More than 100,000 LoC (Not « HelloWorld » Toys)
  - Answer to *Is the Stale Reference pathology so frequent ?*
- Choices : SW using Services
  - JOnAS, Sling, SIP Communicator, Newton
  - *Remark: some use (partially) Component Models*
  - *Remark: Eclipse (Extension Points) & GlassFish (HK2 comp.) are not pertinent !*
- And the results are …

Experiment results

I	OSGi-based software	**JOnAS** (JavaEE server)	**SIP Comm.** (multiprotocol VoIP and Chat UA)	**Newton** (SCA container)	**Sling** (Content Repository)
II	Version	5.0.1	Alpha 3	1.2.3	2.0 incubator snapshot
III	OSGi Impl.	Felix 1.0	Felix 1.0	Equinox 3.3.0	Felix 1.0
IV	Bundles using Component Models	20 iPOJO	6 Service Binder	0	18 Declarative Services
V	Lines of Code	Over 1 500 000	Aprox. 120 000	Aprox. 85 000	Over 125 000
VI	Total Bundles	86	53	90	41
VII	Initial No. of Service Refs.	82	30	142	105
VIII	No. of Bundles w/ Stale Svcs.	4	17	25	2
IX	**No. of Stale Services Found**	**7**	**19**	**58**	**3**
X	**No. of Stale Threads**	**2**	**4**	**0**	**0**
XI	**Stale Services Ratio (IX/VII)**	**8.5 %**	**63 %**	**40.8%**	**2.8%**

[1] Actually the whole Newton implementation is an SCA constructed on top of OSGi, but its bundles did not use an OSGi component model like the other analyzed applications did.

Stale References are not a myth !

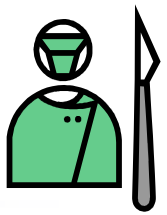
Conclusion

- Stale References are not a myth !
- But Component Models are helpful !
  - JOnAS bundles that used a component model (iPOJO) did not present stale references
  - Same for Sling
  - SIP Communicator errors were mostly due to GUI objects retaining references, and services kept as class attributes
  - Newton does not used identified OSGi component model …

Perspectives

- Release ServiceCoroner in an OSGi OSS Community
- Collaborations to improve current OSGi-based SWs
  - JOnAS but others are welcome
- Add other pathologies diagnostics to ServiceCoroner
  - “Stale” extension points
    - Eclipse IDE & RCP’ plugins
  - Other “stale pathologies” related to the R4.1’ Extender Model
    - HK2, SCA …

More about the ServiceCoroner



- Kiev Gama and Didier Donsez. Runtime Diagnosis of Stale References in the OSGi Services Platform. Technical Presentation at the OSGi Community Event, Berlin, Germany, June 10-11, 2008.
- Kiev Gama and Didier Donsez. Service Coroner: A Diagnostic Tool for Locating OSGi Stale References. In: Proceedings of the 34th EUROMICRO Conference on Software Engineering and Advanced Applications, Parma, Italy, September 3-5, 2008.

- Demos, documentations and tools available on
  - http://www-adele.imag.fr/users/Kiev.Gama/dev/osgi/servicecoroner

  Or googlize "ServiceCoroner"

- Extra stuff : JConsole & VisualVM Plugins for OSGi
  - Bundle admin, Felix/Equinox/KF remote shells, …
  - http://www-adele.imag.fr/users/Didier.Donsez/dev/osgi/jconsole.osgi/

Short demo !

Q & A