



UNIVERSITE
JOSEPH FOURIER
SCIENCES. TECHNOLOGIE. MEDECINE



Towards Dynamic Component Isolation in a Service Oriented Platform

Kiev Gama and Didier Donsez

Université Joseph Fourier

LIG Laboratory, ADELE Team

Firstname.lastname@imag.fr

Agenda

- Motivations
- Isolation Mechanisms in Java
- Our approach
- Perspectives

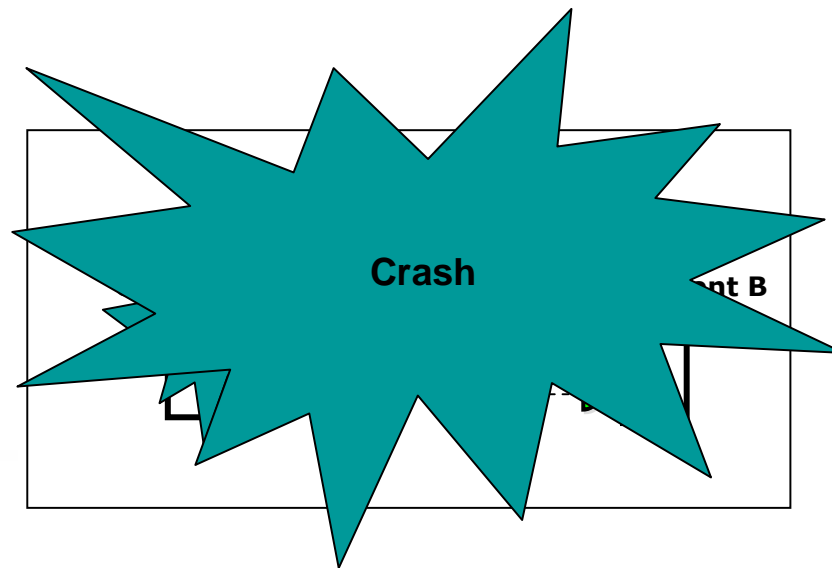
Motivations

- Component based applications **dependability**
- Components Trustworthiness = Measured and perceived dependability (reliability+safety+robustness+availability+security) [Schmidt]
- Provide some sort of isolation for preventing fault propagation
- Not necessarily fault tolerance IN components
- Fault containment to protect underlying application

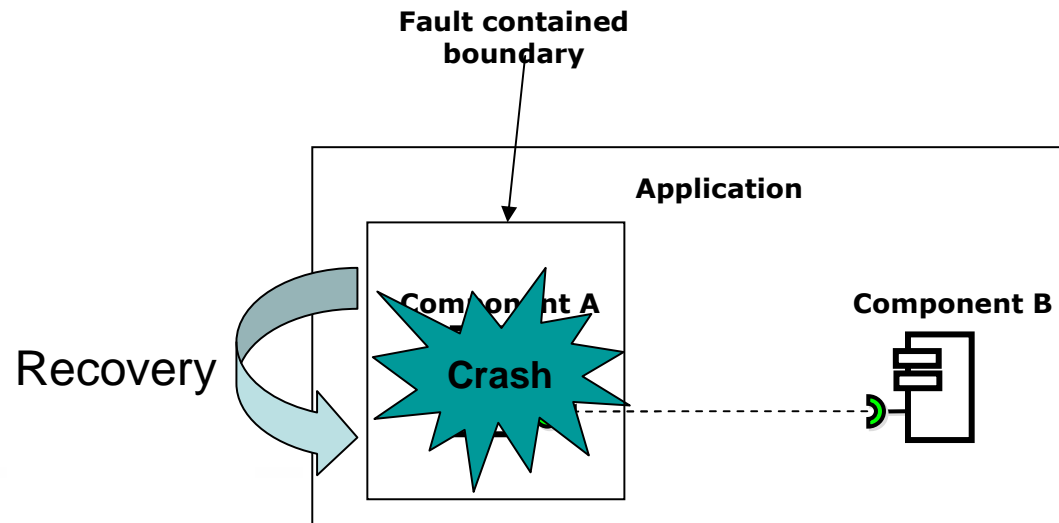
Why?

- “Strength of a composition is defined by its weakest component” [Szyperski]
- We can’t easily predict and test all possible compositions
- Worse in dynamic platforms: we can not even predict what assembly will be deployed
- Need to execute untrustworthy (not necessarily malicious) components but still ensuring system reliability

What we don't want to have



What we would like to have



How?

- Isolating the component with a fault contained boundary

But...

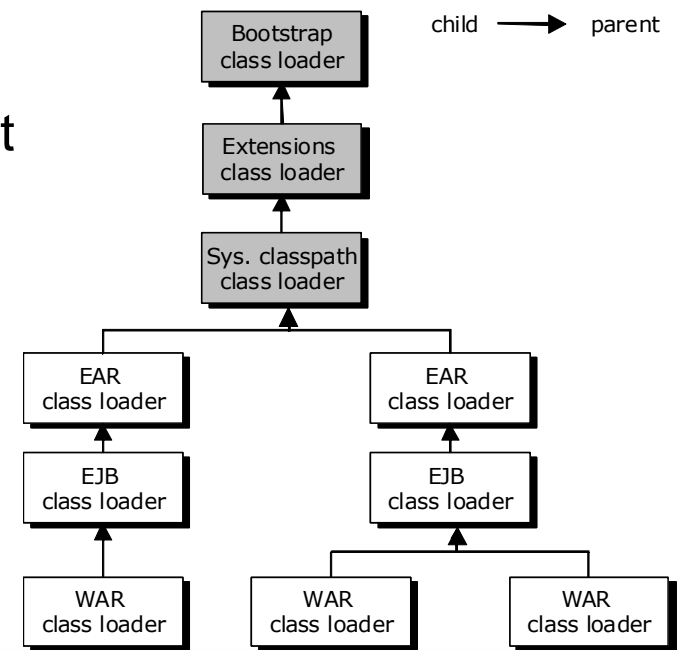
- What isolation mechanisms are available in Java?

Isolation Mechanisms in Java

- Namespace-based
- OS-based
- Domain Isolation

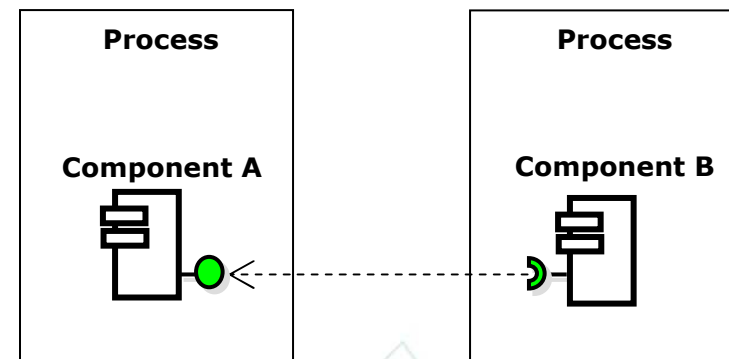
Namespace-based Isolation

- Class loader hierarchy enforcing type isolation
- Pseudo-isolation = No fault containment



OS-based Isolation

- Uses processes as boundaries
- Implies inter-process communication (IPC) costs
- Memory footprint

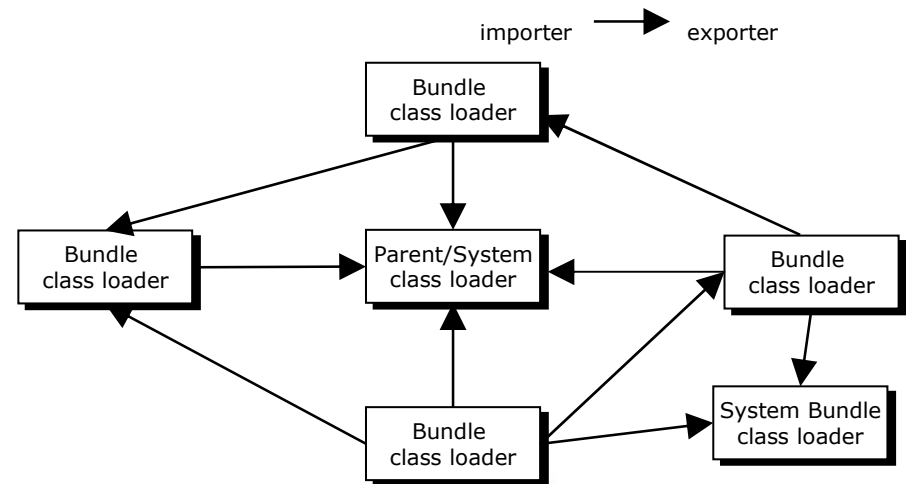


Domain Isolation

- A standardized mechanism: Java Isolates (Java Specification Request 121)
- Fault contained entities
- Applications can be executed in an Isolate
- No widespread adoption (yet) in the Java Platform (concepts applied to Java Micro Edition)

Isolation in the OSGi platform

- Rather a graph than an simple hierarchy of class loaders
- Enhances Java's namespace based isolation
- Finer grain of control on types visibility



OSGi's dynamicity

- **Loose component decoupling through services**
- **Dependencies:**
 - Defined at development time
 - Resolved at runtime
- **Components may be installed and uninstalled during application execution**
- **Weak isolation: memory leaks when components are uninstalled (our previous work)**
- **No fault isolation (just enhanced namespace-based isolation)**

Our Approach

- A sandbox architecture for untrustworthy OSGi components
- A policy for sandboxing
- Dynamically changing of the policy
- Initial prototype based on Isolates
 - Patched Apache Felix 1.4.0
 - SunLabs MVM (Multitasking Virtual Machine) with Isolate API

Prototype

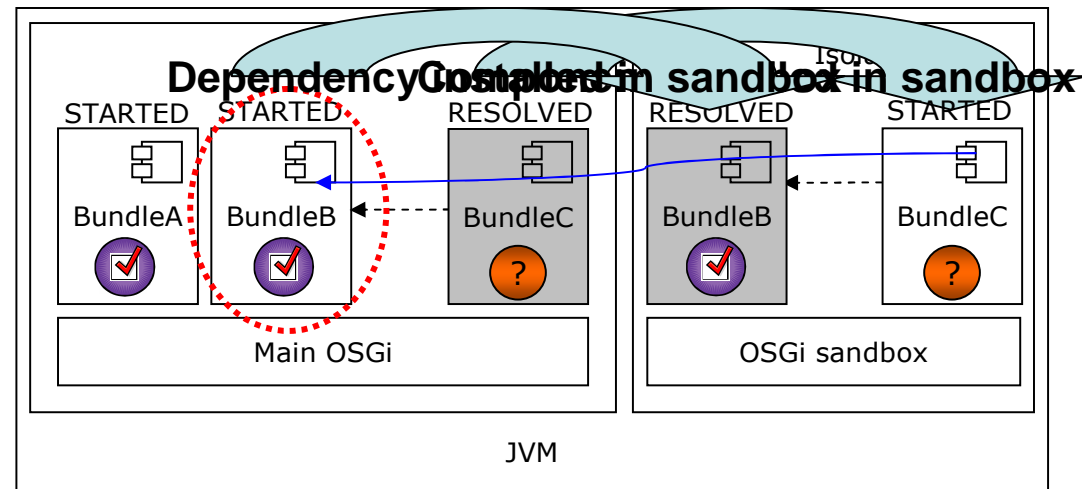
- **Two OSGi frameworks executing in separate Isolates**
 - **Main OSGi**
 - **Sandbox OSGi**
- **Policy defines how the trustworthiness is verified**
- **Untrustworthy components execute in the sandbox**

Prototype Details

- **Communication between platforms done with Links (from the Isolate API)**
 - Not so performing...
- **Dynamic Java proxies on the consumer side**
- **Search strategy:**
 - If service not found on local registry, search “neighbor”
 - main queries sandbox and vice-versa (no cycles though)
- **Attempt to make transparent calls (no java.rmi.Remote)**
 - For the moment only methods using primitive types and Strings

Untrustworthy Component installation example

Services invocation will go through inter-isolate communication



Reset strategy

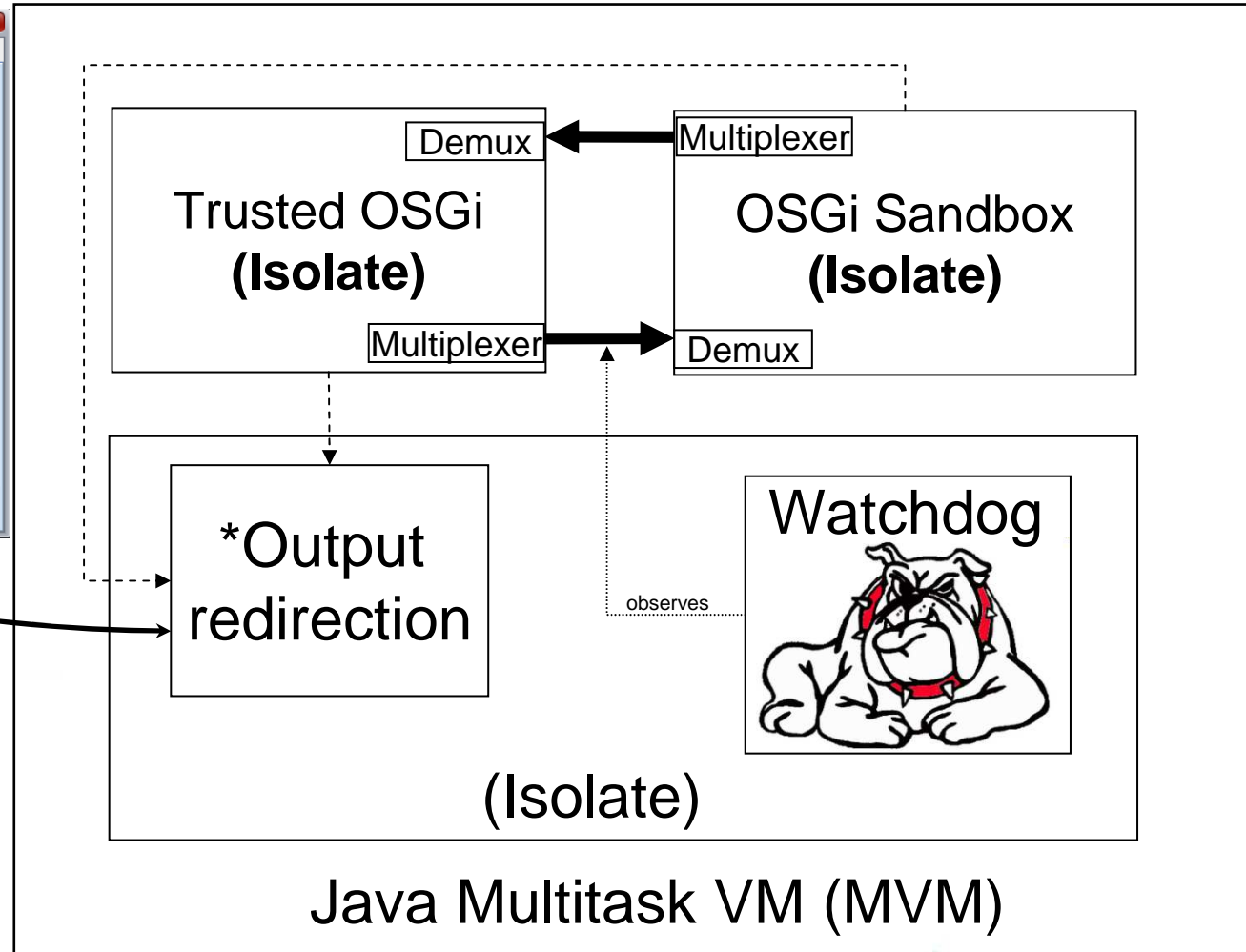
- Good strategy for fixing intermittent fails
- Not something new
 - Microreboot
 - Fault recovery (remedy)
 - Faulty components reboot
 - Software rejuvenation
 - Fault prevention

Watchdog

*Output redirection in separate tabs

```

START LEVEL 1
ID State Level Name
[ 0] [Active] [[ 0] System Bundle (1.4.0)
[ 1] [Active] [[ 1] Apache Felix Shell Service (1.0.2)
[ 2] [Active] [[ 1] Apache Felix Shell TUI (1.0.2)
[ 5] [Installed] [[ 1] file:///export/home/kiev/dev/osgi/org.osgi.servic
[ 8] [Resolved] [[ 1] Foo Example (1.0.0)
[ 9] [Resolved] [[ 1] JACBench (0.1.0.SNAPSHOT)
[10] [Installed] [[ 1] Bench Impl (0.1.0.SNAPSHOT)
[11] [Active] [[ 1] BeanShell Bundle (1.0.0)
    
```



Légende

.....> Listener

---> Socket Connection

→ Communication via Link API (MVM specific)

Tested Functionalities

- Intentional crashes for testing purposes
 - Automatic reboot of sandbox
 - Main platform did not stop
- Stale Services from main platform
 - Bad referrers are in the sandbox isolate
 - Reboot « kills » them
- Component 'promotion'
 - However no state migration

Limitations

- One sandbox only
- Communication protocol
- Inconsistency for `getBundle()` on Isolated Service Reference
- No fine grained resource monitoring

Conclusions and Perspectives

- A fault contained component sandbox is feasible
- Isolation comes at a cost
- Resource accounting is needed for better control
- For the near future:
 - Extending the mechanism for inter-VM communication

[Merci|Obrigado|Gracias|Thanks]

