EL4J: Introduction

an mini introduction presentation

by Philipp H. Oser

26.10.2005

TECHNOLOGY CONSULTING INNOVATION

ELCA

Agenda

- Overview
- Build system: EL4Ant
- Module abstraction
- Remoting
- Other features

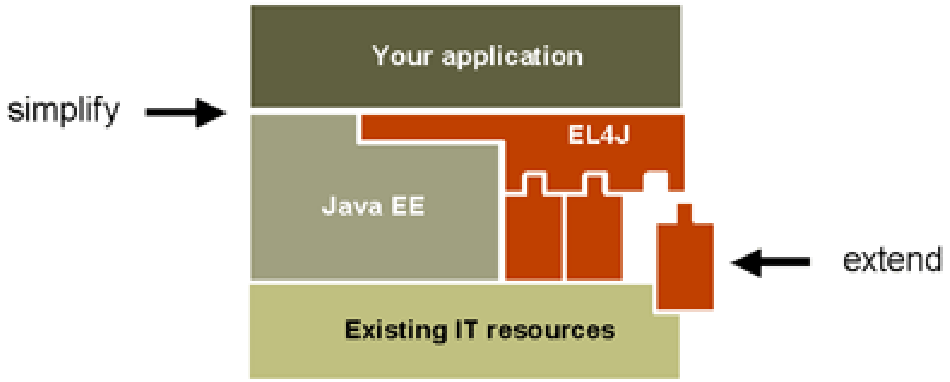
EL4J

EL4J is the Enterprise Library for the J2EE

It adds incremental enhancements to the Spring Java framework

Initial publication: October 2005

Used in 10+ projects



http://el4j.sourceforge.net/

The new build system: EL4Ant (1/2)

Manages development, build, and run tasks

Uses standard Ant (generates a standard build.xml file)

Module abstraction

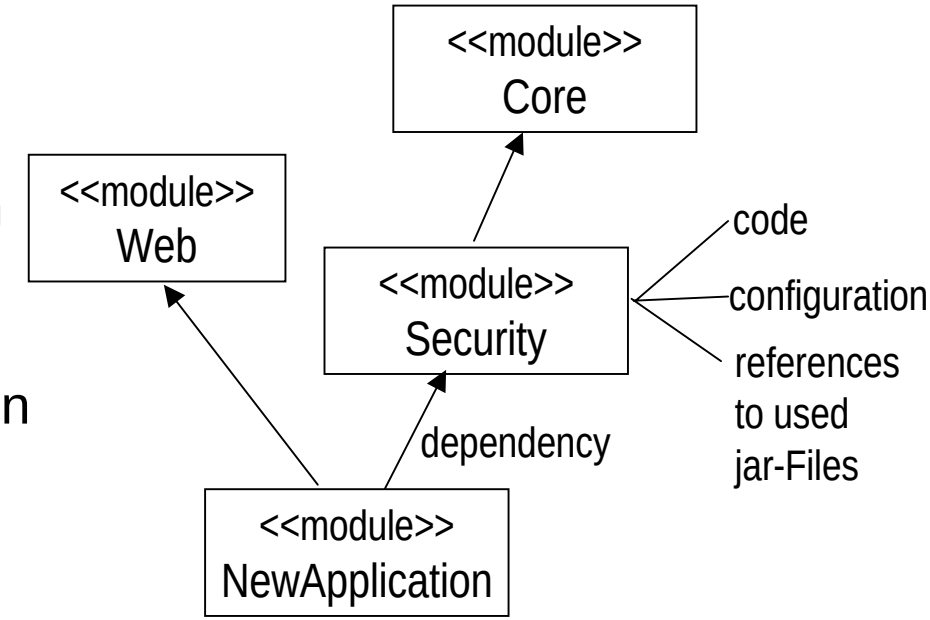
- Organizes code and configuration
- Manages *transitive* dependencies
- Dependencies are taken into account, when launching operation on a module
- Binary and source modules

Light & usable w/o EL4J

Plugin-extendable

Published under
http://el4ant.sourceforge.net/

Sample use of modules:



The new build system: EL4Ant (2/2)

Plugins (basic)

- Compile, run, website
- Javadoc: generate javadoc
- Checkstyle: verify checks on code
- Junit: auto test execution and report
- Emma: code coverage
- J2EE-war: make and deploy war files, control web container
- Eclipse plugin: use from within eclipse, with dependencies

Newer features

- Distributed tests: tests with more than 1 JVM, reuses junit plugin
- Multi-Environment support: say start.web instead of start.tomcat
- ear support
- EJB support (in collaboration with ejbRemoting)
- Manage SQL scripts
- Macker: check code usage rules (e.g. for layering enforcement)

Module abstraction

EL4J brings the module abstraction of EL4Ant on the level of spring

- One can split applications in modules
- Each module can have its own code, configuration and dependencies

Rationale:

- Modularity: organize work in smaller sub-parts, to simplify & separate development
- Provide default configuration for modules: reduce complexity
- Dependency management (1): each module lists its requirements (other modules and jar files). These dependencies are then automatically managed (downloaded if needed, added to the classpath, added to deployment packages such as WAR, EAR or zip files)
- Dependency management (2): only resources of the dependent modules are visible (you can e.g. hide server-side jar files during compilation of client-side code, ensure they are not used)

Module abstraction

How is the module abstraction implemented?

- the module abstraction of EL4Ant (the build system)
- the ModuleApplicationContext (a light wrapper for the standard Spring application context, this is optional)
- a convention on how to organize configuration information within each module

Remoting features

POJO remoting with SOAP and EJB

- Don’t write a EJB session bean facade by hand for POJOs

Implicit context passing (optional)

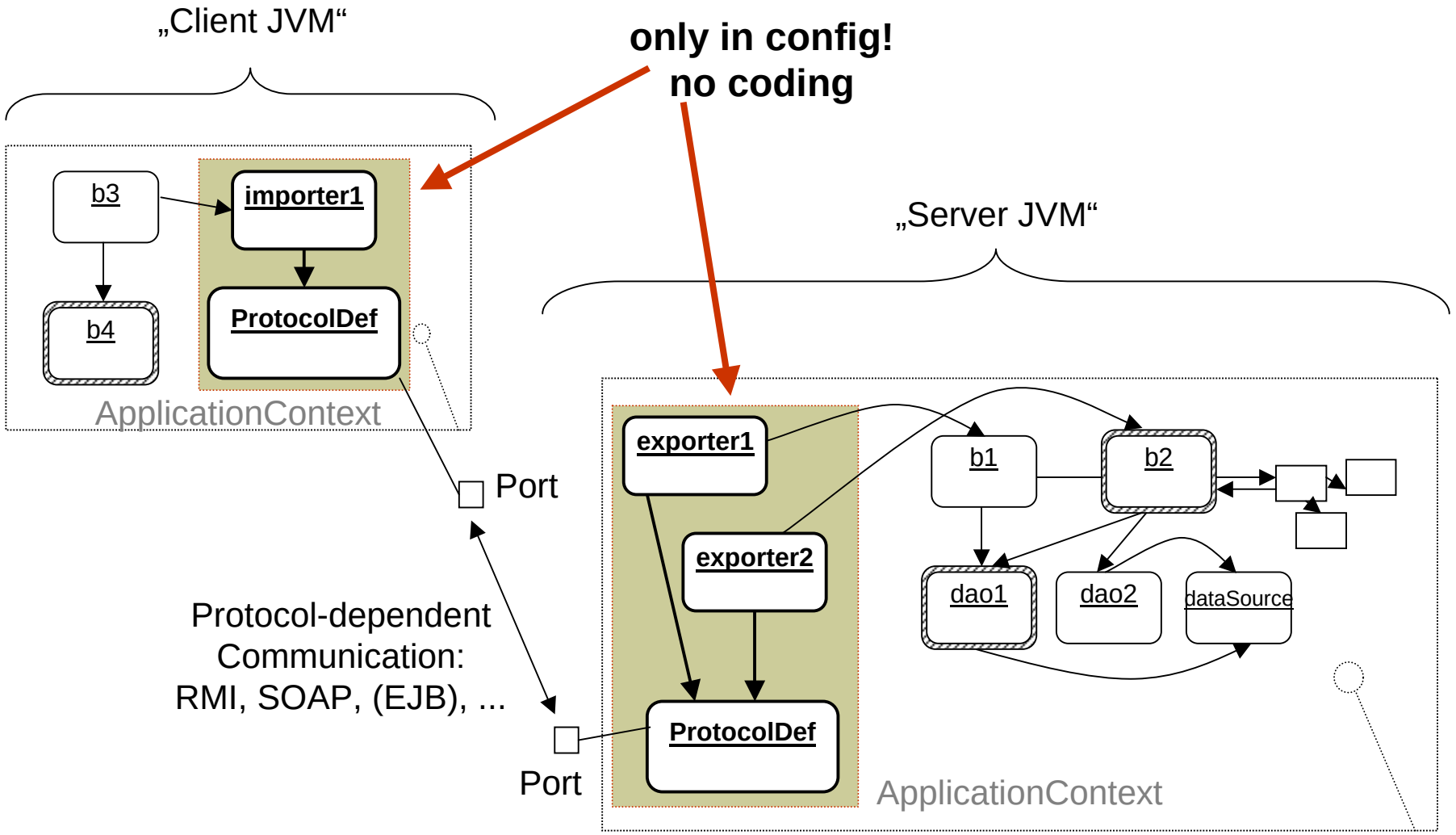
- Add implicit information to remote calls -- transparent for the user
- Sample usage: security context, performance measurement data, identity on whose behalf we make a request, …

Simplification in the remoting configuration

- Make switching between protocols easier

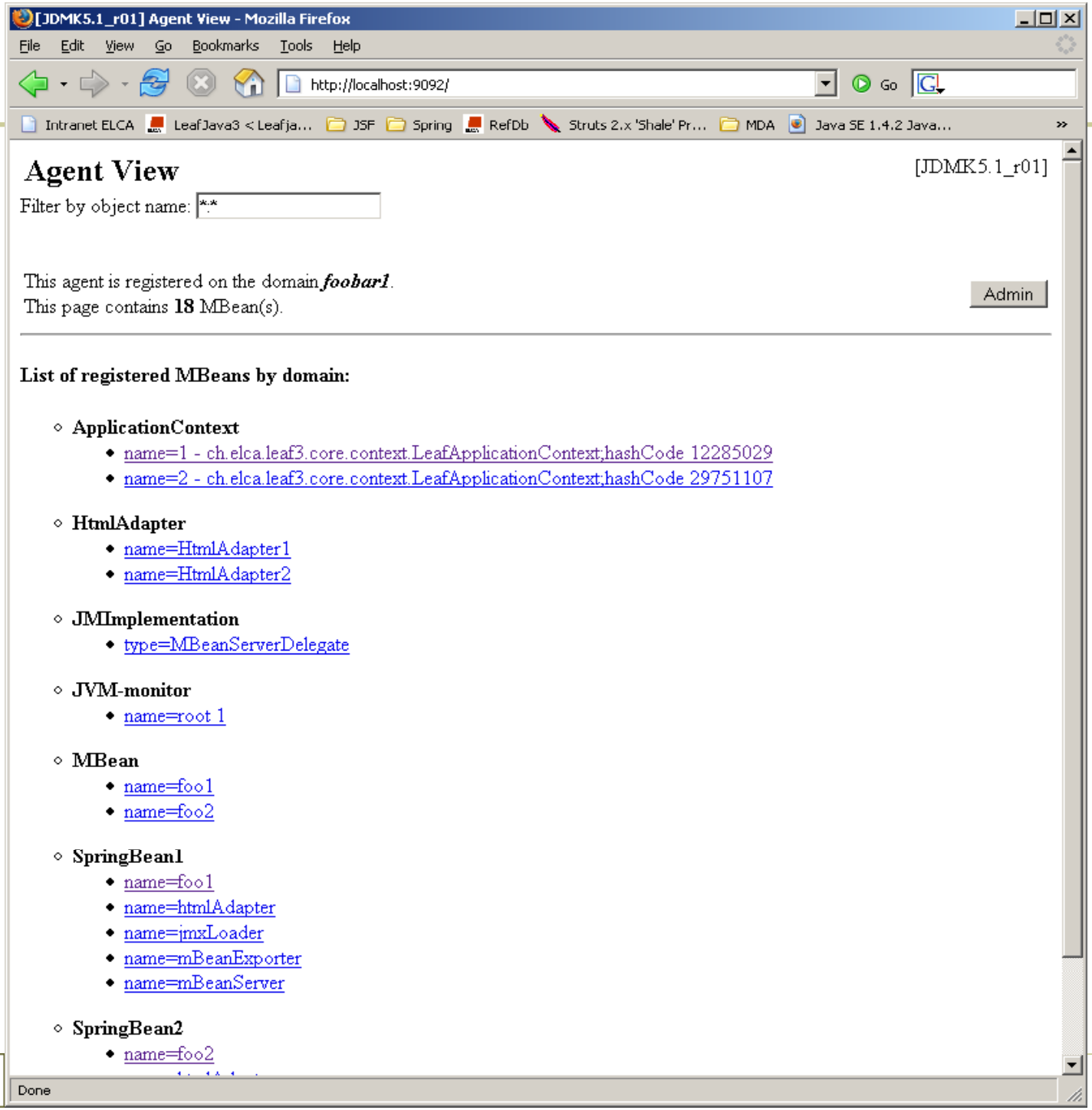
Documentation of remoting semantics

New features: Remoting

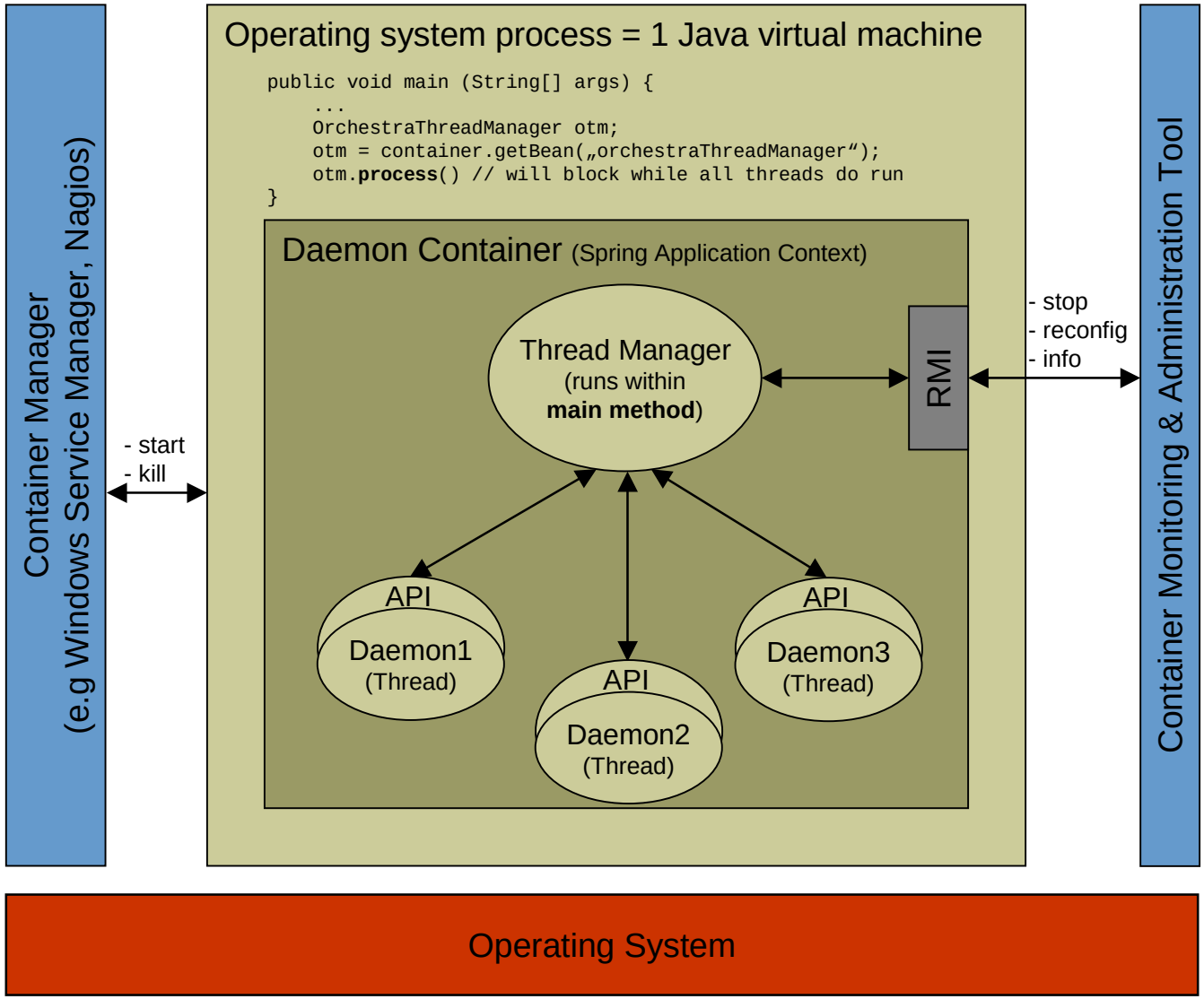


Other features

- No-effort publication of all spring beans and their config to JMX
- Daemon manager to supervise and manage long-running daemons
- An exception handling framework implementing a safety façade
- Various little helpers for the Spring core
- Documentation and guidelines


[JDMK5.1_r01] Agent View - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:9092/
Go
Intranet ELCA
LeafJava3 < Leafja...
JSF
Spring
RefDb
Struts 2.x 'Shale' Pr...
MDA
Java SE 1.4.2 Java...
Agent View
[JDMK5.1_r01]
Filter by object name: *:*
This agent is registered on the domain foobar1.
This page contains 18 MBean(s).
Admin
List of registered MBeans by domain:
ApplicationContext
name=1 - ch.elca.leaf3.core.context.LeafApplicationContext;hashCode 12285029
name=2 - ch.elca.leaf3.core.context.LeafApplicationContext;hashCode 29751107
HtmlAdapter
name=HtmlAdapter1
name=HtmlAdapter2
JMImplementation
type=MBeanServerDelegate
JVM-monitor
name=root 1
MBean
name=foo1
name=foo2
SpringBean1
name=foo1
name=htmlAdapter
name=jmxLoader
name=mBeanExporter
name=mBeanServer
SpringBean2
name=foo2
Done

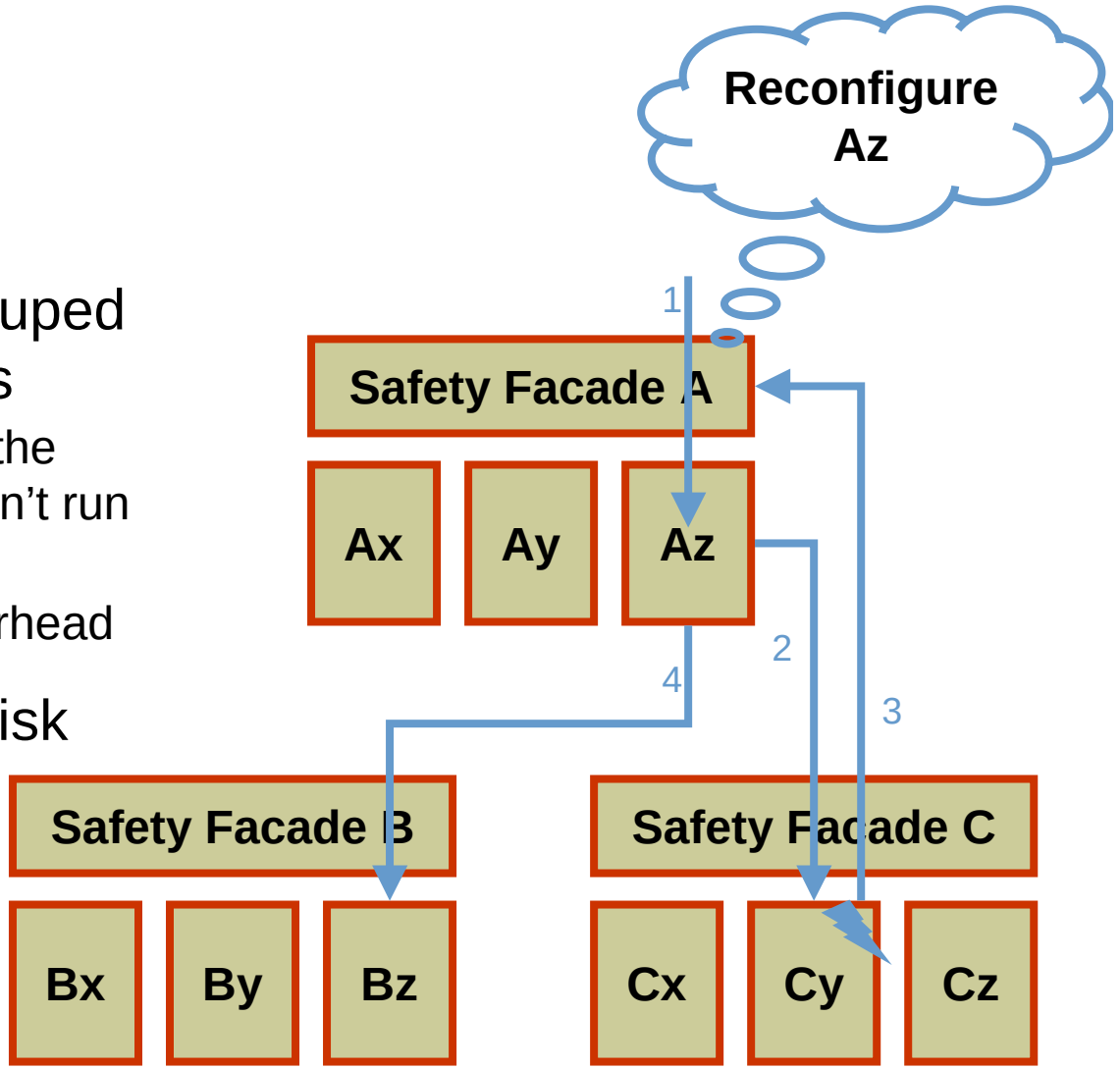
Daemon manager



Intelligent exception handling - Safety Facade

Safety Facade – Risk Communities

- Safety façade handles technical exceptions for “normal code”
  - Separation of concern
- Components can be grouped to build risk communities
  - Accessing components of the same risk community doesn’t run through the safety facade
  - Reduces performance overhead
- Hierarchies of different risk communities
  - Flexible configuration
  - Autonomic reconfiguration



References

http://el4j.sourceforge.net/

http://www.elca.ch/

info@elca.ch

Thank you for your attention

For further information please contact:

Philipp H. Oser
Manager
+41 44 456 32 11
Philipp.Oser@nospam.elca.ch

Christian Gasser
CTO
+41 21 613 21 11
christian.gasser@nospam.elca.ch

TECHNOLOGY CONSULTING INNOVATION

ELCA