Jakub Wozniak
G. Kruk, S. Deghaye
BE/CO/AP CERN, Geneva, Switzerland

ICALEPCS'09
Kobe, Japan
14/10/2009

NEW WAVE OF COMPONENT REUSE WITH SPRING FRAMEWORK AP CASE STUDY

Agenda

- Spring Framework
- Dependency Injection and IoC Pattern
- Design for Reuse
- CERN Examples
- Lessons learned
- Questions

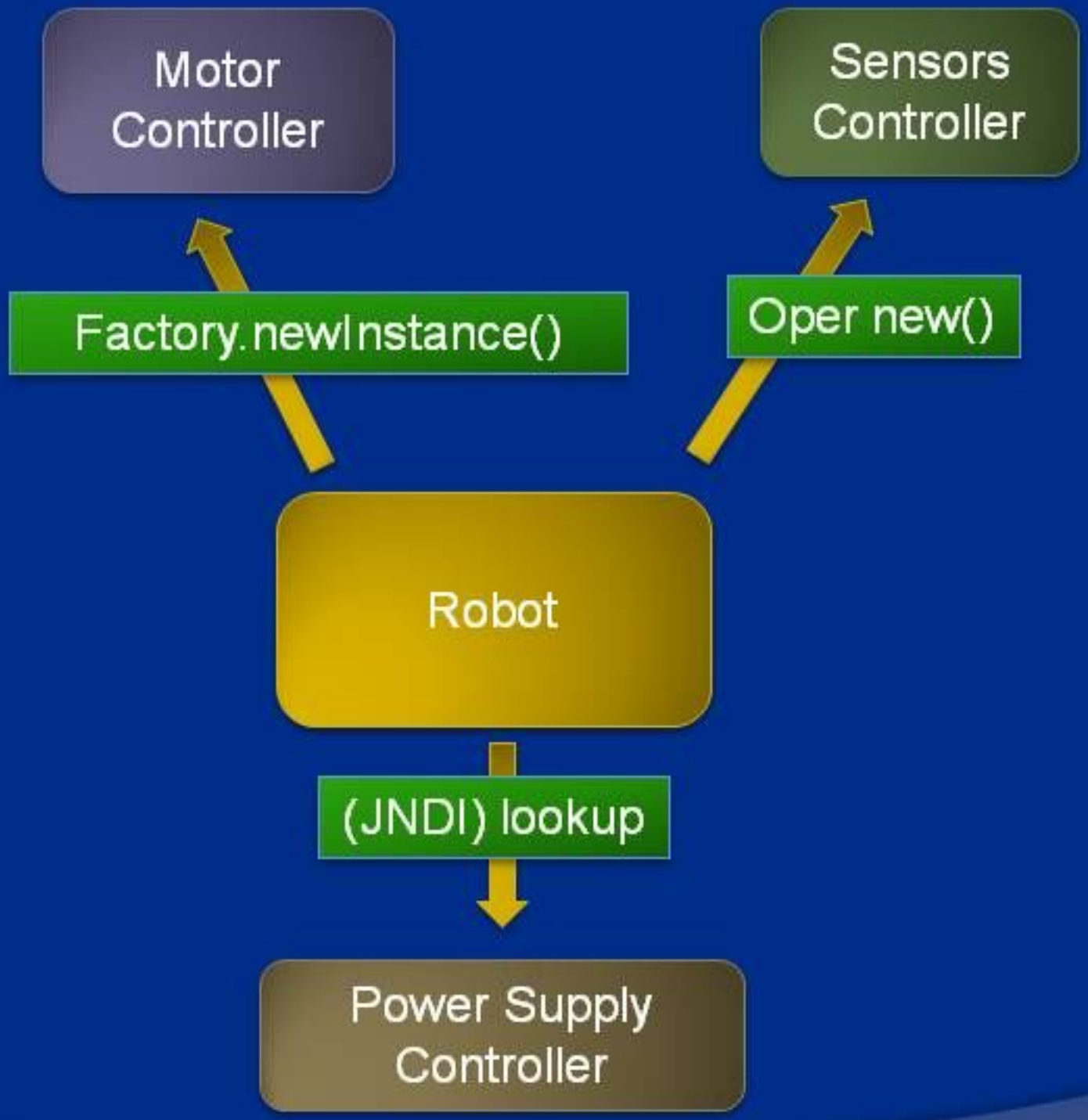
Spring Framework

- OpenSource framework existing since early 2003 for Java and .NET
- Designed to make the entreprise development easier
  - Non-intrusive Dependency Injection patterns
  - Only POJOs – just simple classes called beans
  - Used as a glue code
  - Allows to focus on business operations
- Largely modular architecture allows using only specific parts
- Promotes robustness, extensibility and reusability by nature

Dependency Injection

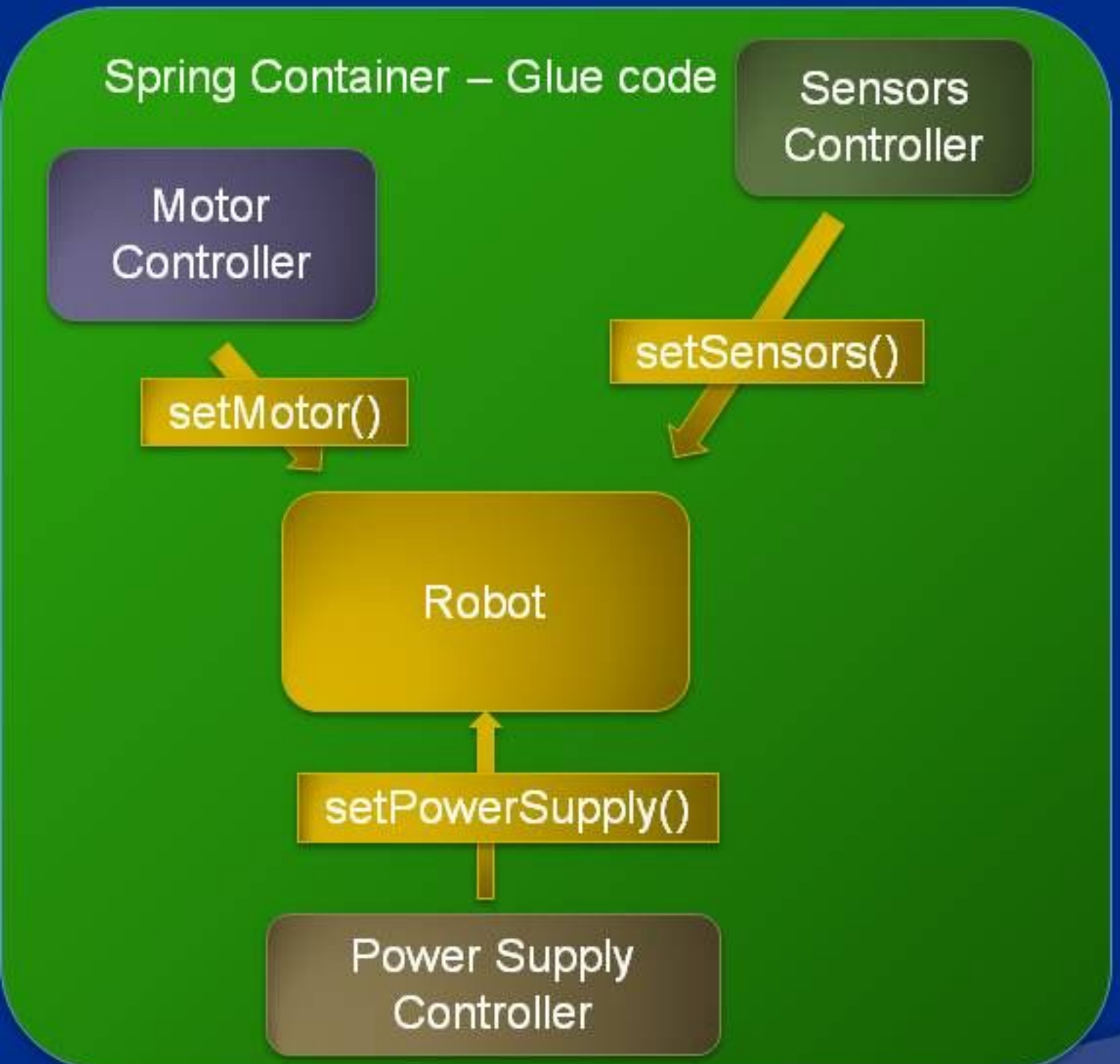
- How objects get their dependencies
- Just simple setters or constructor arguments
- Dependencies are managed and "injected" by container during runtime
- aka Inversion of Control or Hollywood Principle (don't call us we will call you).

Old Style Dependency Management



- Object has to find its dependencies ifself
- Strong coupling between objects
- Mix of business and framework plumbing code
- Difficult to change implementation
- Difficult to test
- Difficult to reuse

New Dependency Injection Style

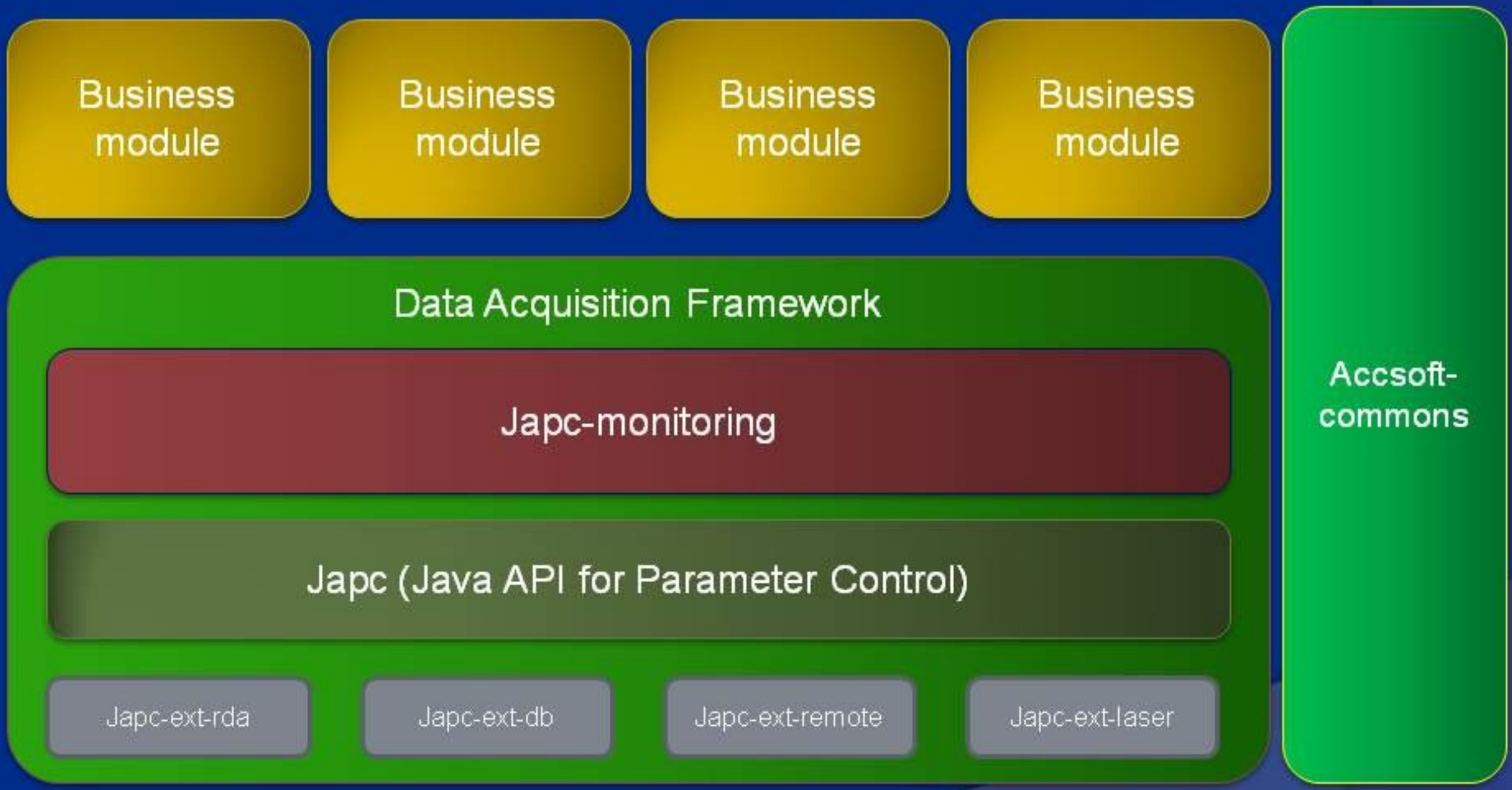


- Prevents hard-coded object creation or service lookup
- Loose coupling between objects
- Only business code
- Helps write effective unit tests with mocks
- Easy reuse of components!

Design Principles

- Use interfaces, hide the implementation
  - Easy testing, easy integration
  - They constitutes components
- Create your domain objects – common language between different systems
  - No problems with eventual integration
- Use similar solutions everywhere
  - Easy code takeover and maintenance
  - Less dependencies
- Extract generic code from the rest
  - Maybe you will reuse it somewhere else so…
  - Keep it clean!

CERN Examples



CERN Examples

Fixed Disp

DB Logging Services

Software Interlock em

Accsoft-commons

Data Acquisition Framework

Japc-monitoring

japc

Japc-ext-rda | Japc-ext-db | Japc-ext-remote | Japc-ext-laser

Lessons Learned

- More attention to common language
- Systems developed in cooperation
- Code duplication tracking and removal
  - Smaller code base
  - Less maintenance problems
- Common knowledge base and change management
  - Existing components
  - Changes to libraries
  - Release synchronization and tracking
- More emphasis on software quality standards and testing

Thank you for your attention!

Questions?