

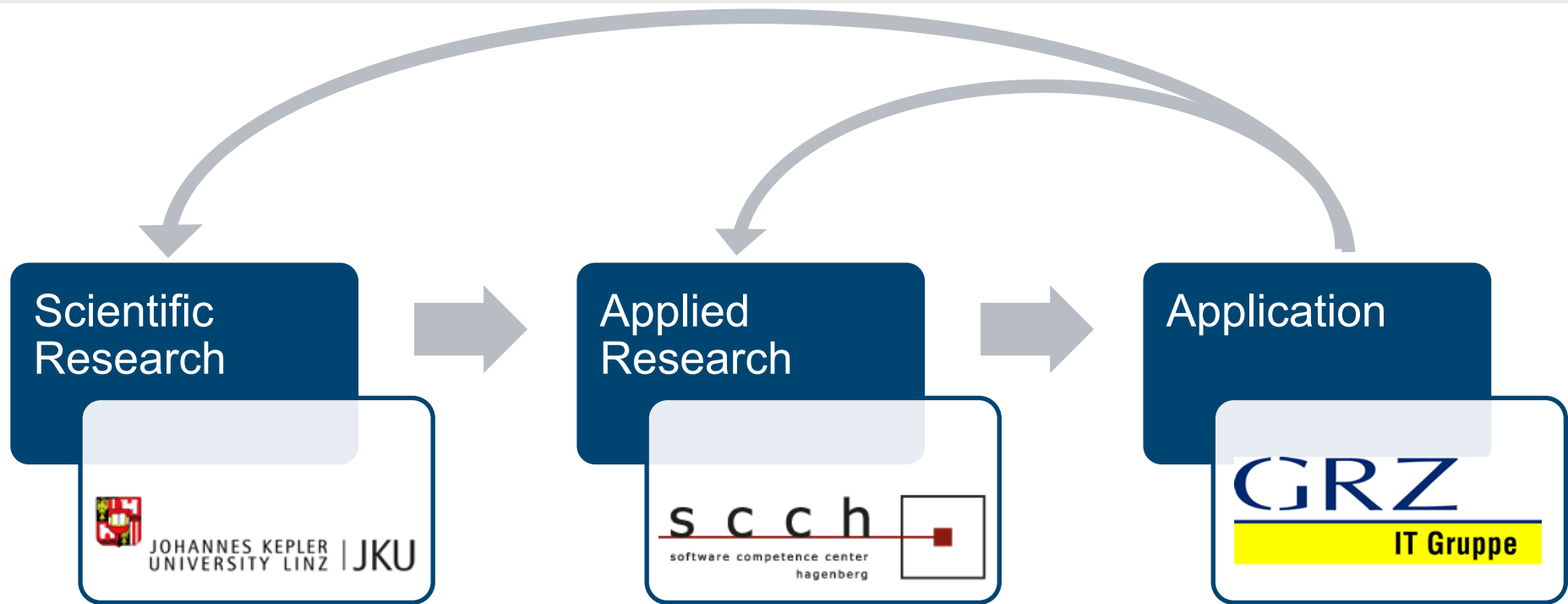
# **Service Development and Architecture Management for an Enterprise SOA**

Thomas Kriechbaum, RACON Software GmbH, Austria

Georg Buchgeher, Software Competence Center Hagenberg, Austria

Rainer Weinreich, Johannes Kepler Universität Linz, Austria

# Setting



- Cooperation on various topics for several years

# GRZ IT Group

- Founded in 1971; employs now more the 780 persons
- One of the major IT-service provider in Austria with the business lines
  - Computing Center
  - Software Development
  - General IT-services
- Comprises three companies
  - GRZ IT Center GmbH
  - RACON Software GmbH
  - PROGRAMMIERFABRIK GmbH
- Raiffeisen Landesbank Oberösterreich AG as general owner



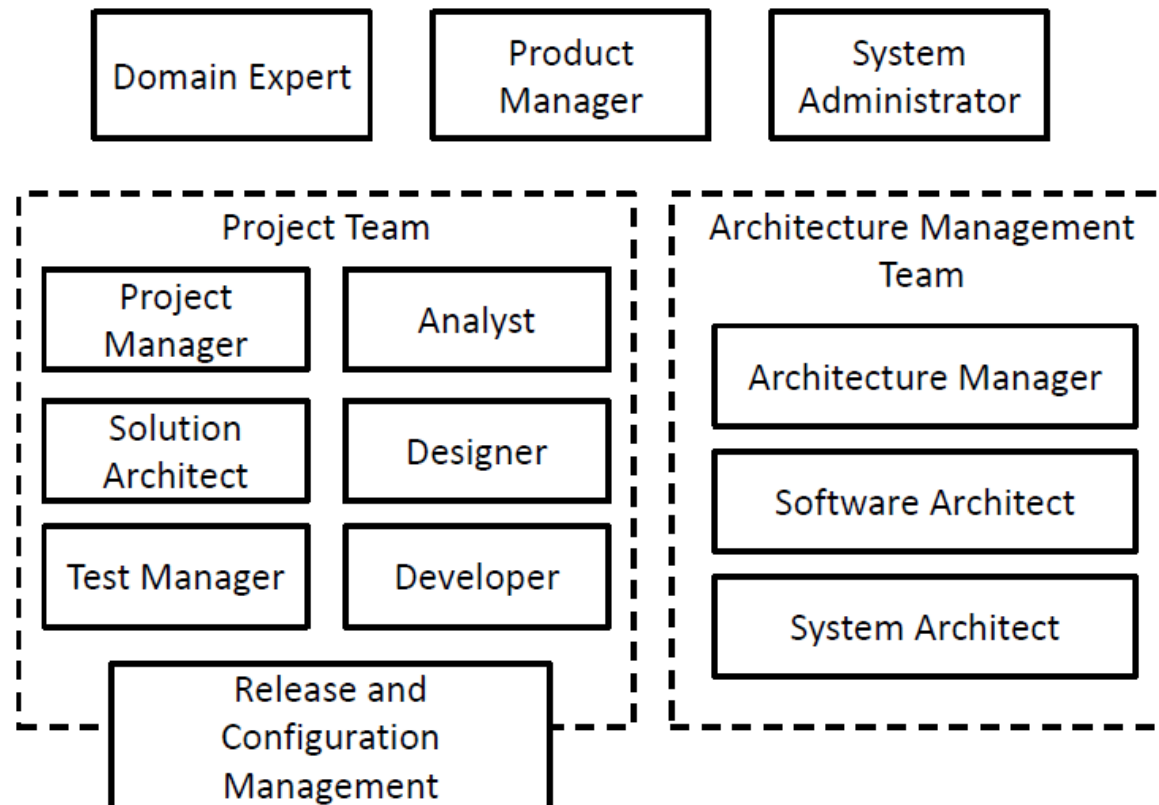
# System Overview

- Enterprise SOA is organized in applications that are clustered in business domains
- Applications are decomposed in modules, which are the unit of versioning and deployment
- Modules have to follow the blueprints of the reference architecture and guidelines of the integration architecture

# System Overview

- Different types of UI-Modules address different communication channels
  - Mobile Apps → end customers
  - Web-Applications → end customers, banking staff
  - Rich Client Applications → business customers, banking staff
- Business logic is primarily implemented in Service- or Mainframe-Modules
  - the core banking system on the mainframe is not treated as legacy system
  - the core banking system is integrated via web-service facades
- A set of infrastructure modules provide cross cutting functionality like security, journaling, monitoring or output management
- 170 Service-Modules with about 1700 services

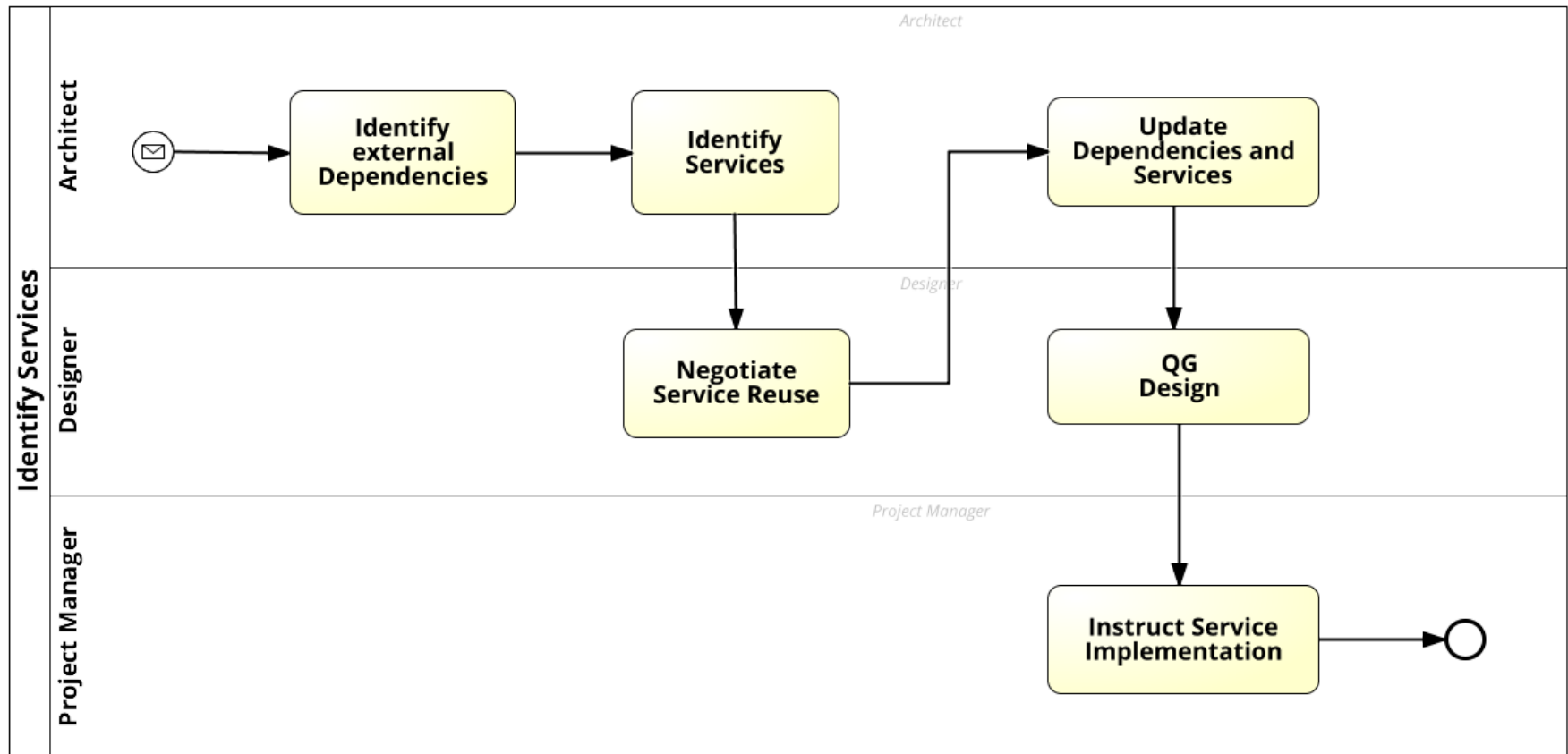
# Stakeholders



# Service Development Process

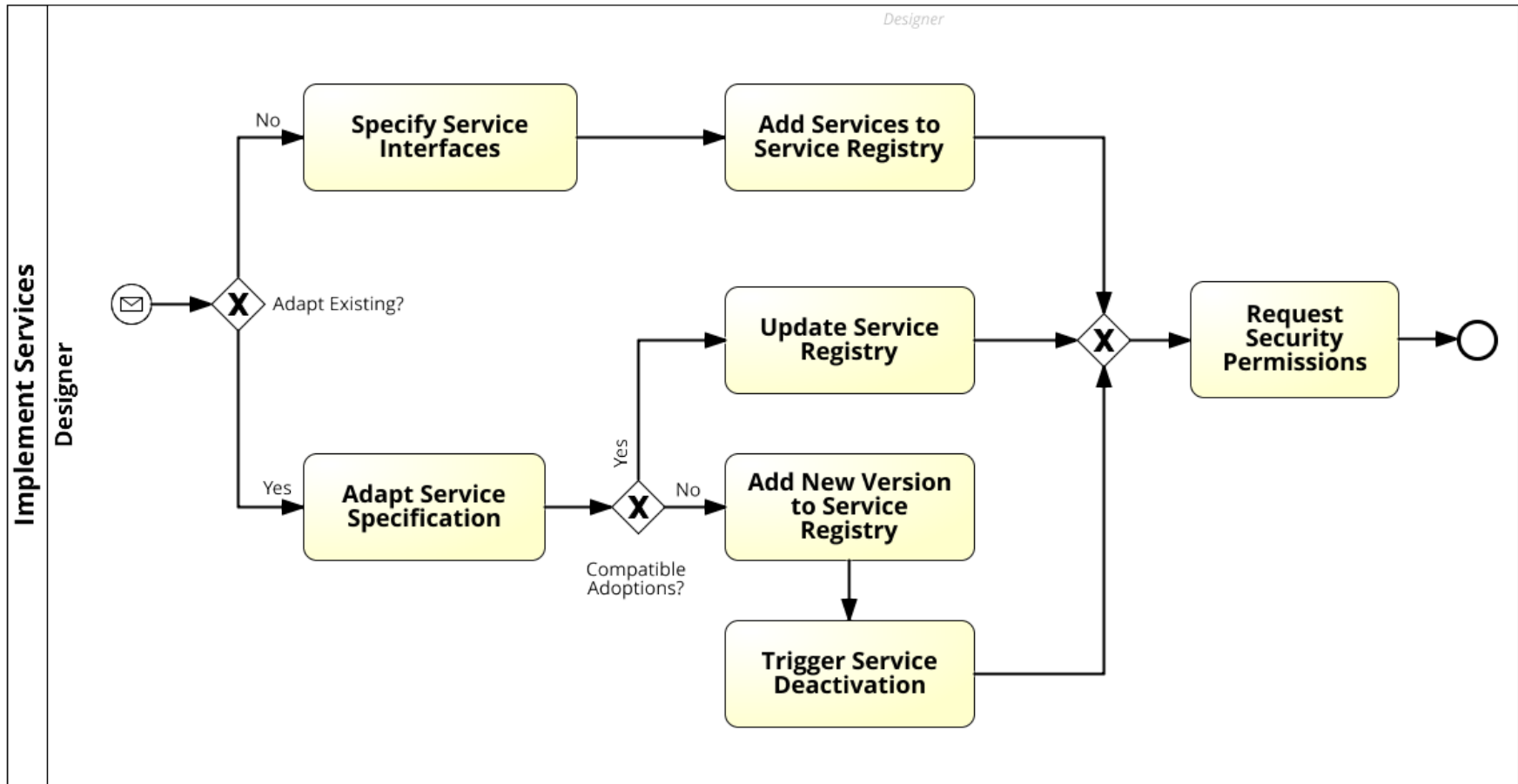
- Embedded in a global product development process
  - Product managers and domain experts gather and prioritize requirements
  - Several projects are set up to implement new product version
  - Project (can) span more architectural layers (e.g. UI, service, mainframe)
- Service-Lifecycle governed by guidelines und directives
  - Service identification
  - Service implementation
  - Service operation and monitoring
  - Service deactivation
- Defined quality gates have to be passed

# Service Development Process

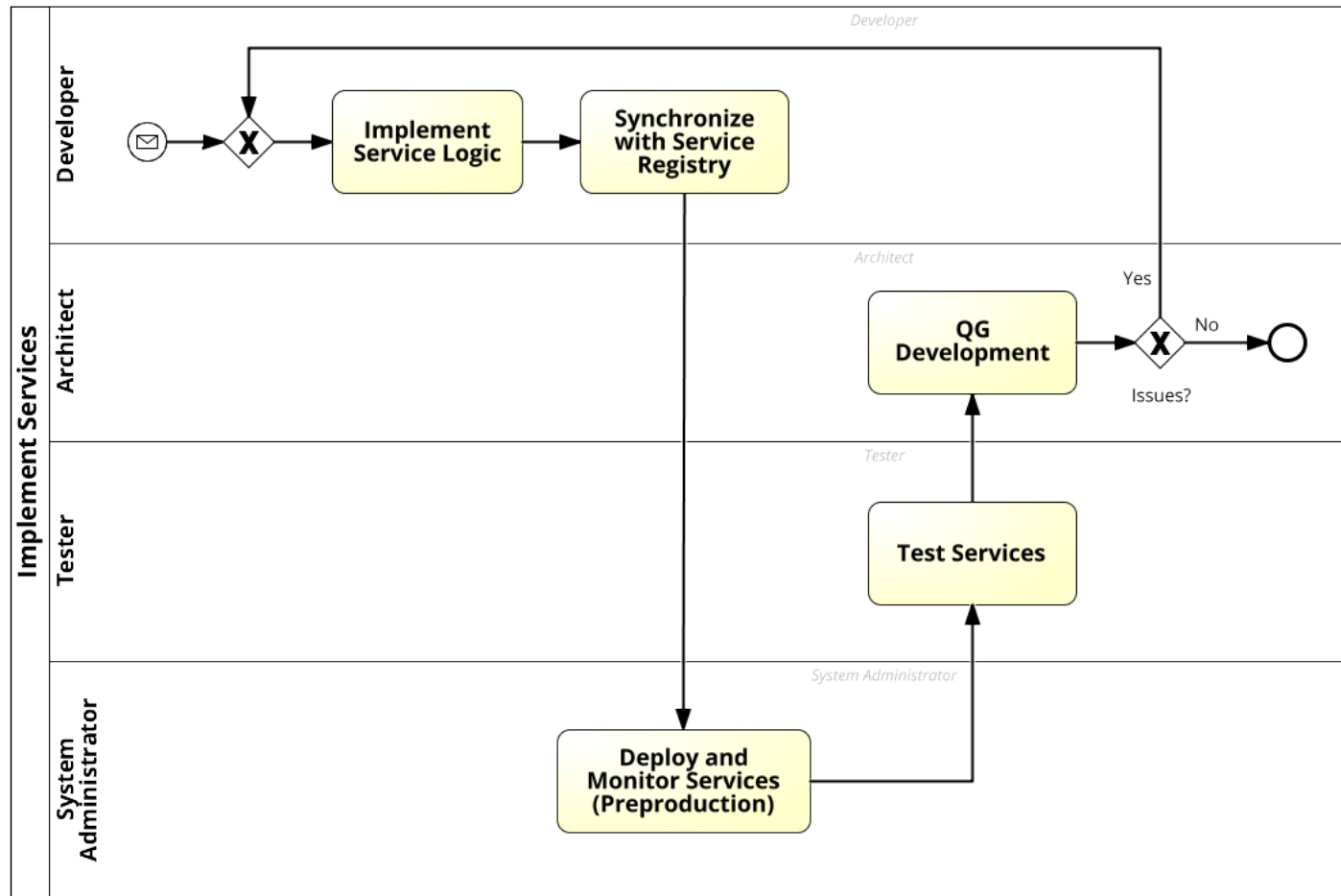




# Service Development Process



# Service Development Process



# Service Technology Stack

## Java Platform 1

since 2003

<b>In-house Framework</b>  Security  Mainframe Integration  Configuration & Composition
<b>Open-Source Frameworks</b>
<b>J2EE 1.4</b>  JNDI, EJB, JMS, JAX-RPC

## Java Platform 2

since 2009

<b>In-house Framework</b>  Security  Mainframe Integration
<b>Spring Core Framework</b>  Configuration & Composition
<b>Open-Source Frameworks</b>
<b>Java EE 5</b>  JNDI, EJB, JMS JCA, JAX-WS

## jRAP-SOA

since 2013

<b>In-house Framework</b>  Security  Mainframe Integration  Configuration & Bootstrapping
<b>Open-Source Frameworks</b>
<b>Java EE 6</b>  JNDI, EJB, JMS, JCA, JAX-WS, JAX-RS, CDI

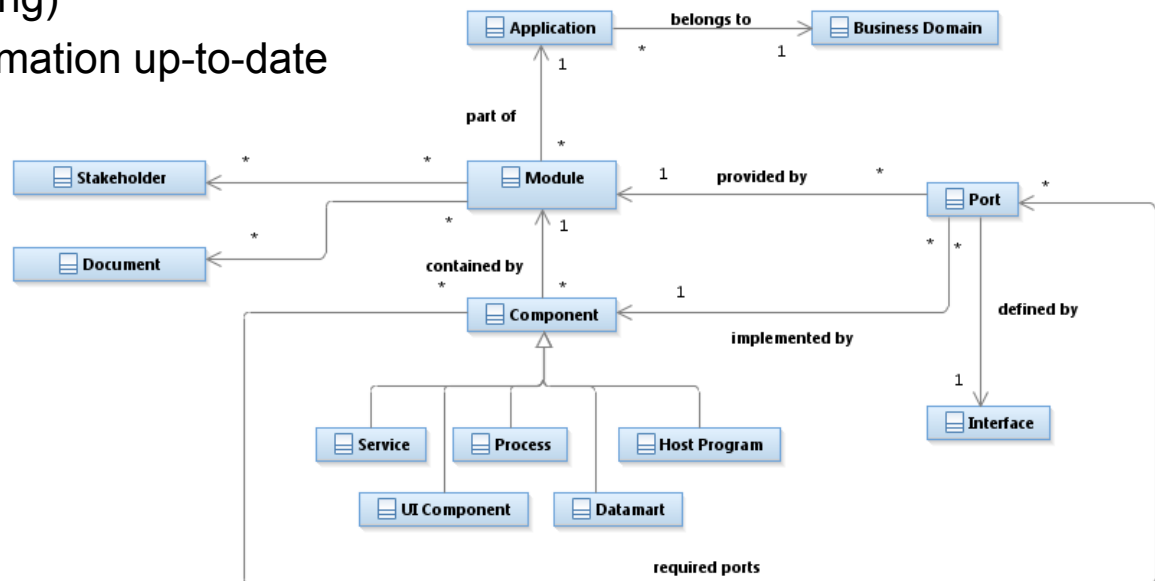
# Service Development Practices

- **Model-Driven Development**
  - Supports top-down-strategy for specifying and implementing services
  - Service-interface and entities are modeled using UML
  - Custom UML-profile and UML-libraries allows to specify additional information
  - Code generation is fully integrated in the Maven-build-process
  - Has been proven to be an important success factor
  
- **Custom Annotations for Architectural Information**
  - CDI-based jRAP-SOA Annotations to classify specific components
  - Allow to control runtime-behavior (Exception-Handling, Security, ...)
  - Are used to extract architectural information

# Service Development Practices

## ■ Service Registry

- Stores information about service-modules, services and dependencies between service-consumers and service-providers
- Information is based on a logical information model to reduce tool and vendor-dependency
- Many different stakeholders with different needs (see project-setting)
- Challenge to keep information up-to-date



# Service Development – Goal



15 min for implementing  
a Web-Service for an  
existing Mainframe-Module

# Service Development – Create Project

**New Maven Project**

New Maven project

Specify Archetype parameters

Group Id: at.jrap.soa.sandbox

Artifact Id: elba-services

Version: 1.0.0-SNAPSHOT

Package: at.racon.elba

Properties available from archetype:

Name	Value
projectCode	ELBASRV
projectName	ELBA-Services
applicationDomain	Vertrieb
applicationShortName	ELBA-Internet
applicationVersion	1.0
moduleShortName	ELBA-Services
moduleVersion	1.0
contextRoot	elba-services_v1_0

Advanced

< Back Next > Cancel Finish

identifying application

identifying module

# Service Development – Model Service Interface

**Properties** **Problems**

**<Attribute> «Property» creditorAccount**

Keywords:

Applied Stereotypes:

Stereotype	Profile	Required	Marking Model
Property	JrapSoaProfile	False	elba-services

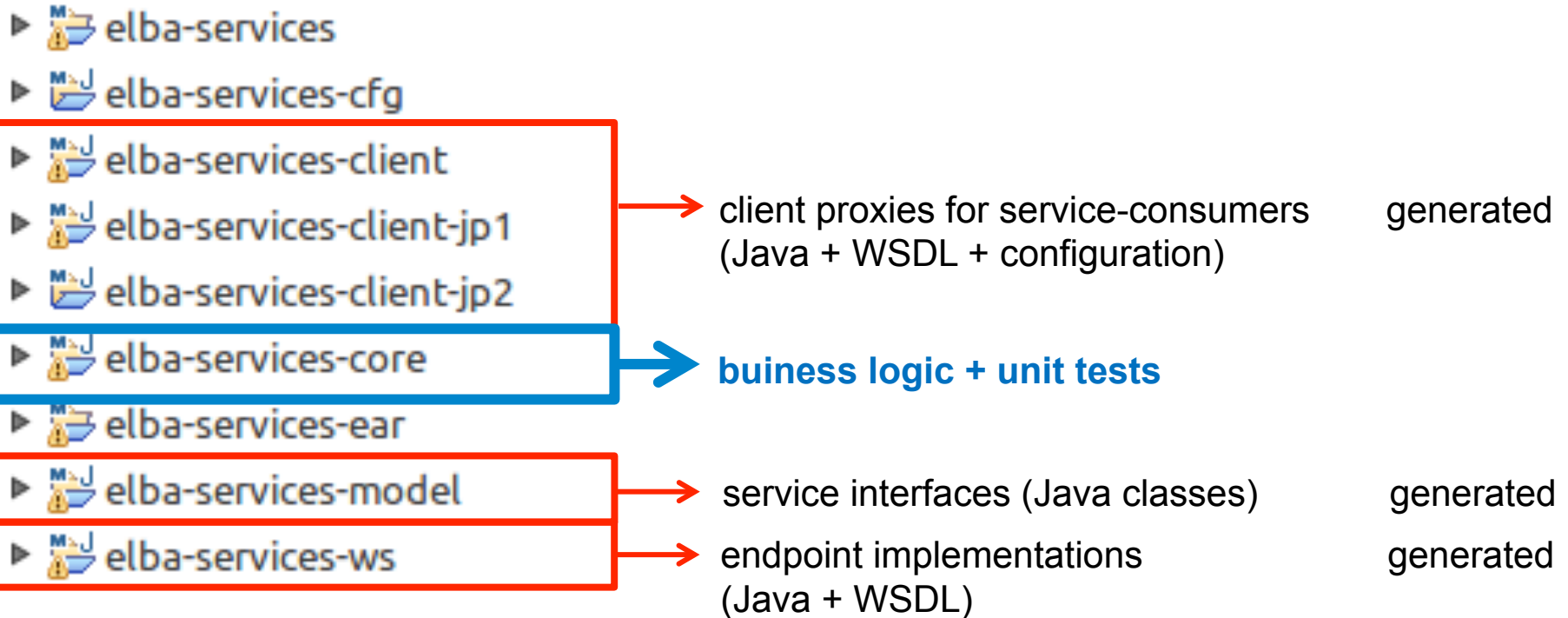
Buttons:

Stereotype Properties:

Property	Value
Property	
JrapSoaProfile::Property::annotations	<input type="button" value="List"/> Entries: 0
JrapSoaProfile::Property::deprecation	
JrapSoaProfile::Property::validation	null
JrapSoaProfile::Property::zsvzType	
nillable	False
transient	False
usage	null



# Service Development – Generate Code



# Service Development – Implement Business Logic

```
@Service
@Stateless
public class PaymentServiceBean extends AbstractBean implements PaymentService {

    private static final long serialVersionUID = 1L;

    @Inject
    private PaymentTransaction paymentTransaction;

    @Override
    @Secured("ELBA.10")
    public void transferDirectDebit(DirectDebit debit) throws PaymentServiceException {
        paymentTransaction.transferDebit(debit);
    }
}
```

# Service Development – Integrate Service

- Add dependency

```
<dependency>
  <groupId>at.jrap.soa.sandbox</groupId>
  <artifactId>elba-services-client</artifactId>
  <version>1.0.0-SNAPSHOT</version>
</dependency>
```

- Configure endpoint address

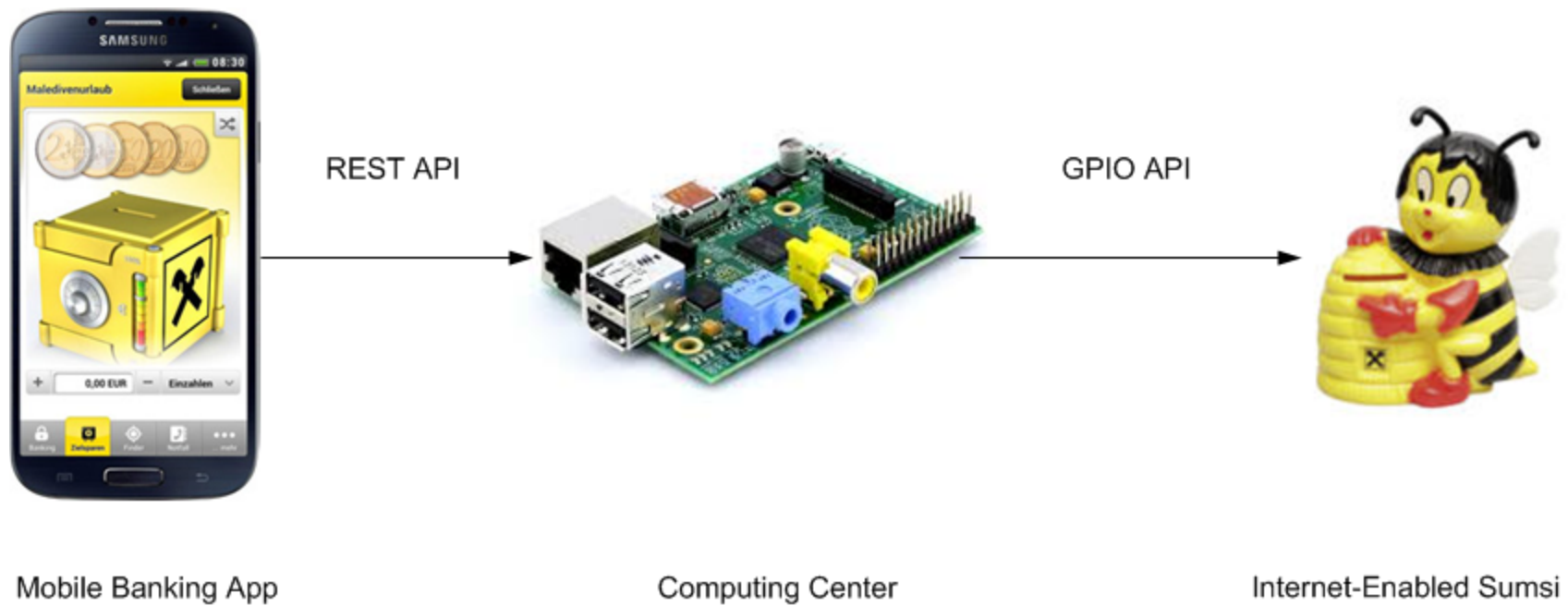
PaymentService.endpointAddress=[http://localhost:9090/elba-services\\_v1\\_0/PaymentService](http://localhost:9090/elba-services_v1_0/PaymentService)

- Inject service and invoke operation

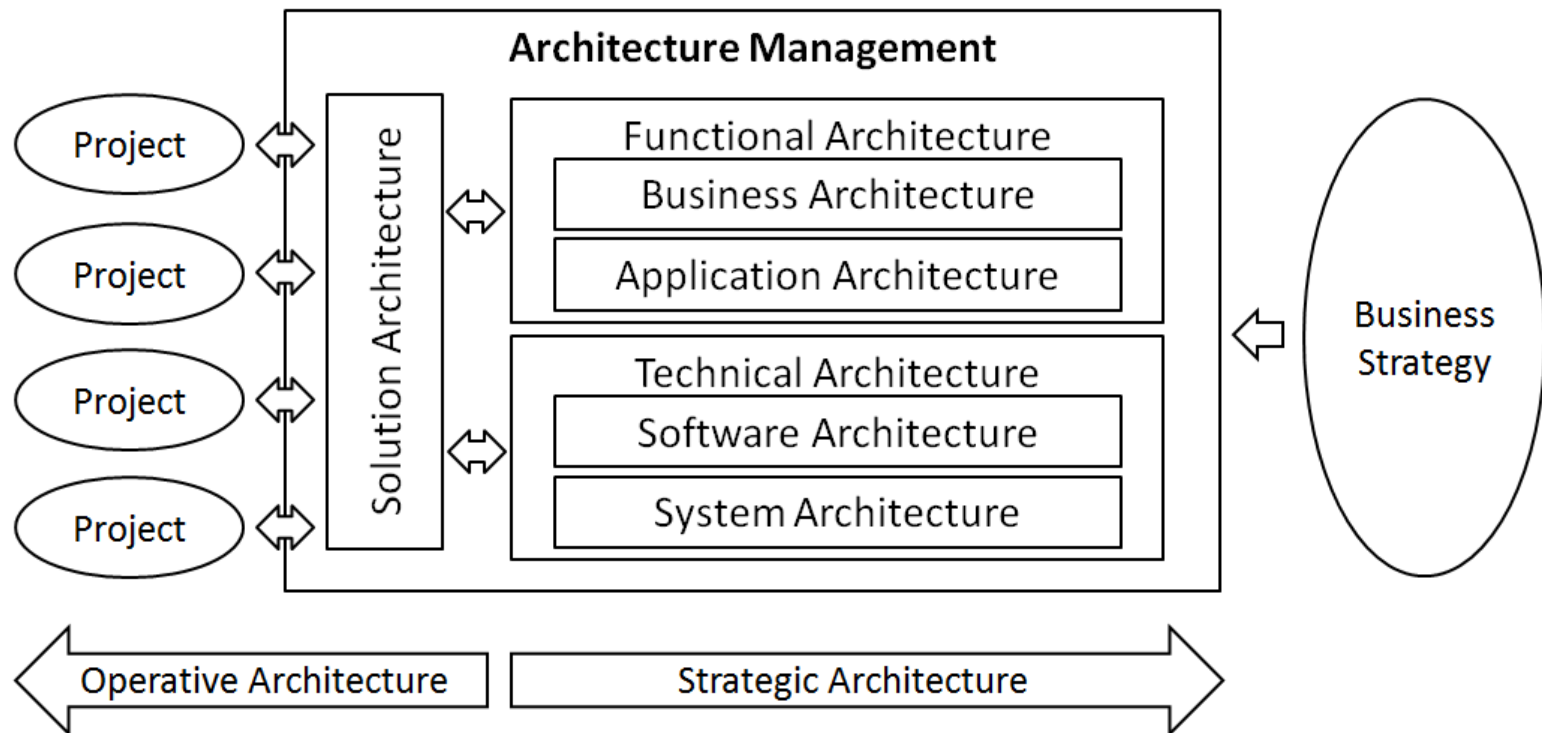
```
@Inject
private PaymentService paymentService;

DirectDebit debit = new DirectDebit();
...
paymentService.transferDirectDebit(debit);
```

# Internet of Things - Strictly Confidential ;-)

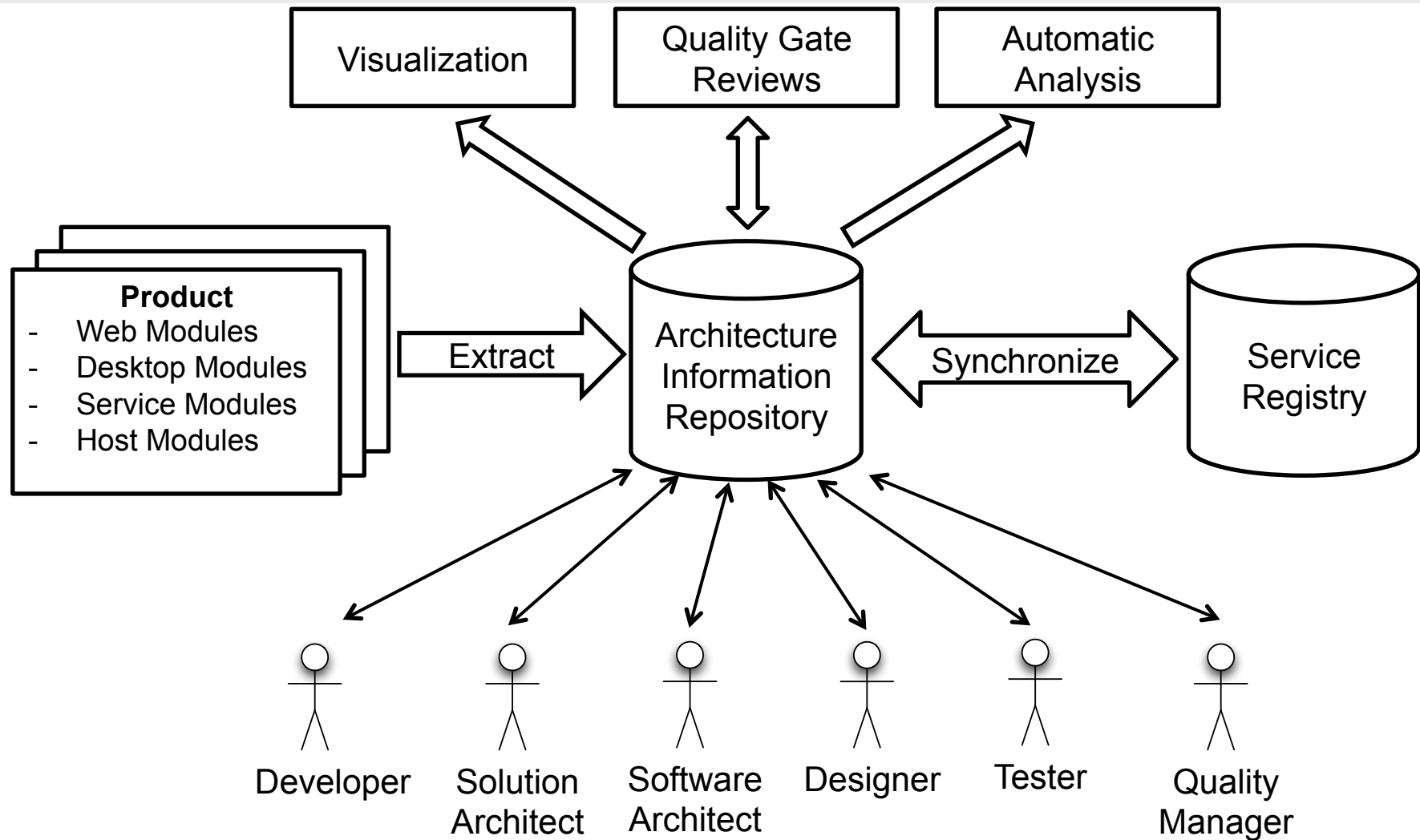


# Architecture Management (Group)



- **Planning:** Definition of EA
- **Development:** Evaluations and Adaptions of EA
- **Controlling:** Design Support and Quality Control

# Overview: Architecture Management Support



# Architecture Extraction & Visualization

- Up-to-date documentation
- Architectural information in implementation
- Incremental architecture extraction
- Simulated component composition
- Architectural views

# Architecture Extraction & Visualization

- Up-to-date documentation
- **Architectural information in implementation**
- Incremental architecture extraction
- Simulated component composition
- Architectural views

```
@Stateless
@Service
public class RegistryServiceBean
implements RegistryService {

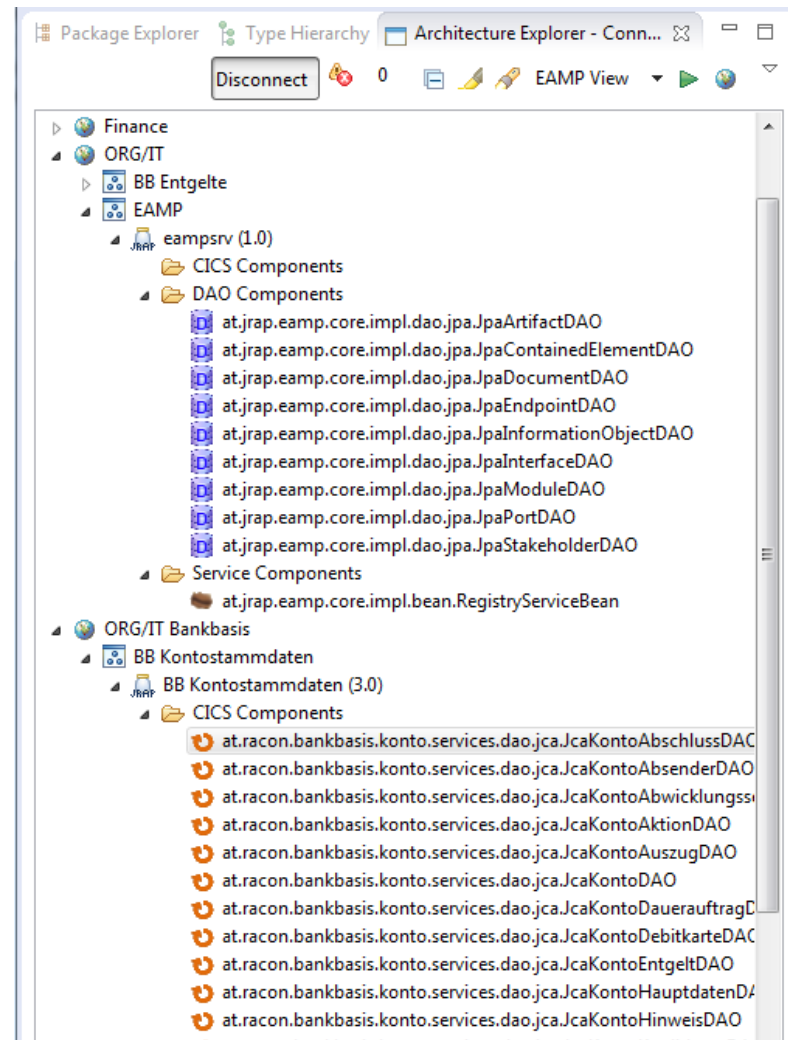
@Inject
private ArtifactConverter converter;
```

```
Manifest-Version: 1.0
Implementation-Vendor: RACON Software GmbH
Application-Domain: ORG/IT
Application-ShortName: EAMP
Implementation-Vendor-Id: RACON Software GmbH
Module-ShortName: eampsrv
Module-Version: 1.0
```



# Architecture Extraction & Visualization

- Up-to-date documentation
- Architectural information in implementation
- Incremental architecture extraction
- Simulated component composition
- **Architectural views**



# Architecture Extraction & Visualization

- Up-to-date documentation
- Architectural information in implementation
- Incremental architecture extraction
- Simulated component composition
- **Architectural views**

The screenshot shows the 'Details' window for a component named 'JcaKontoAbschlussDAO'. The window is divided into several sections:

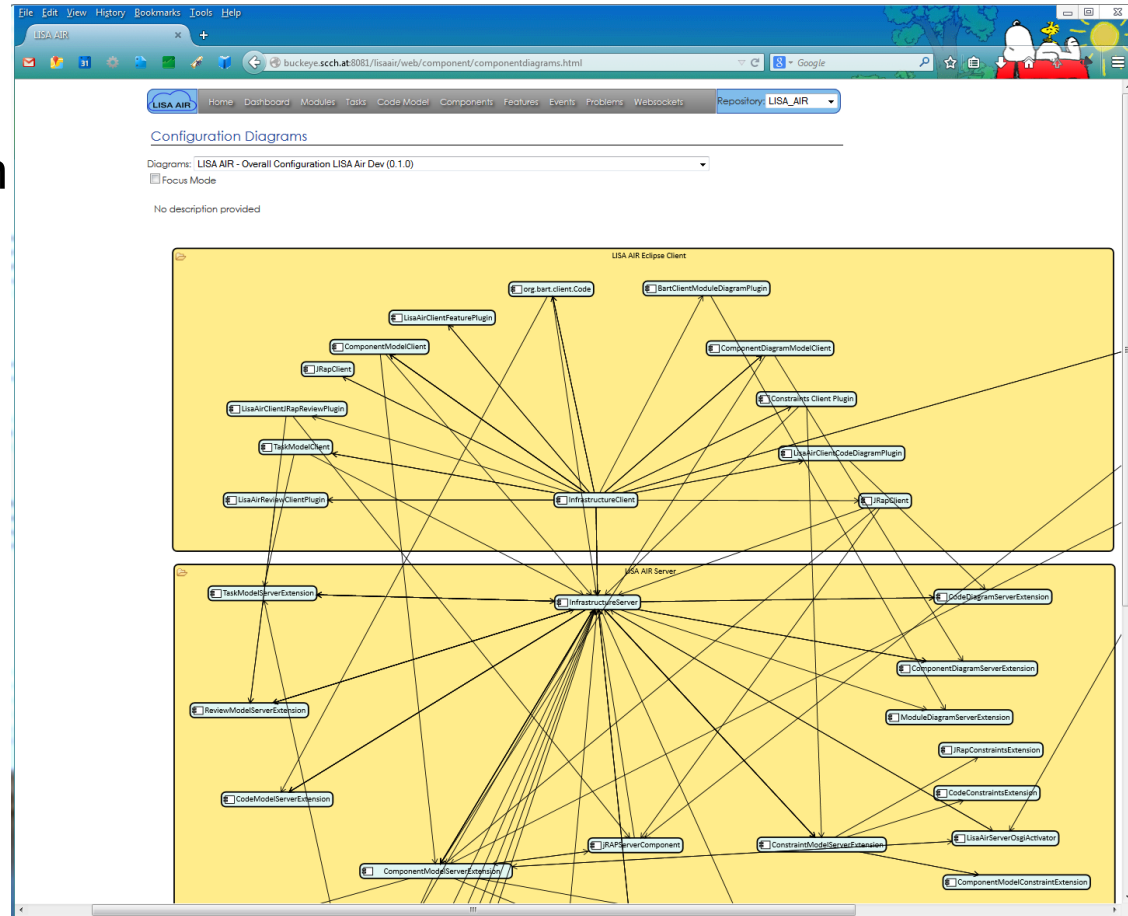
- Component**:
  - Name: JcaKontoAbschlussDAO
  - Binding: JRAP CICS Component
  - Kind: Elementary
  - Documentation: Undefined
- Provided Service(s)**:
  - KontoAbschlussDAO (with a red 'X' icon)
  - ↳ JRAP Bean Port Binding
- Required Service(s)**: (Empty list)
- [Show Component in Architecture Explorer](#)
- Configuration Diagrams**:
  - ↳ [konto-services Module Configuration Diagram](#)
- CICS Component**:
  - Traco Code: B21F, B22F, B38B
- Racon Service Module**:
 

Application-Shortname:	BB Kontostammdaten
Implementation Vendor:	RACON Software GmbH
Implementation Vendor ID:	RACON Software GmbH
Module Shortname:	BB Kontostammdaten
Module Version:	3.0
Application Version:	3.0
Application Domain:	ORG/IT Bankbasis
- Operations**: (Section header, no details visible)



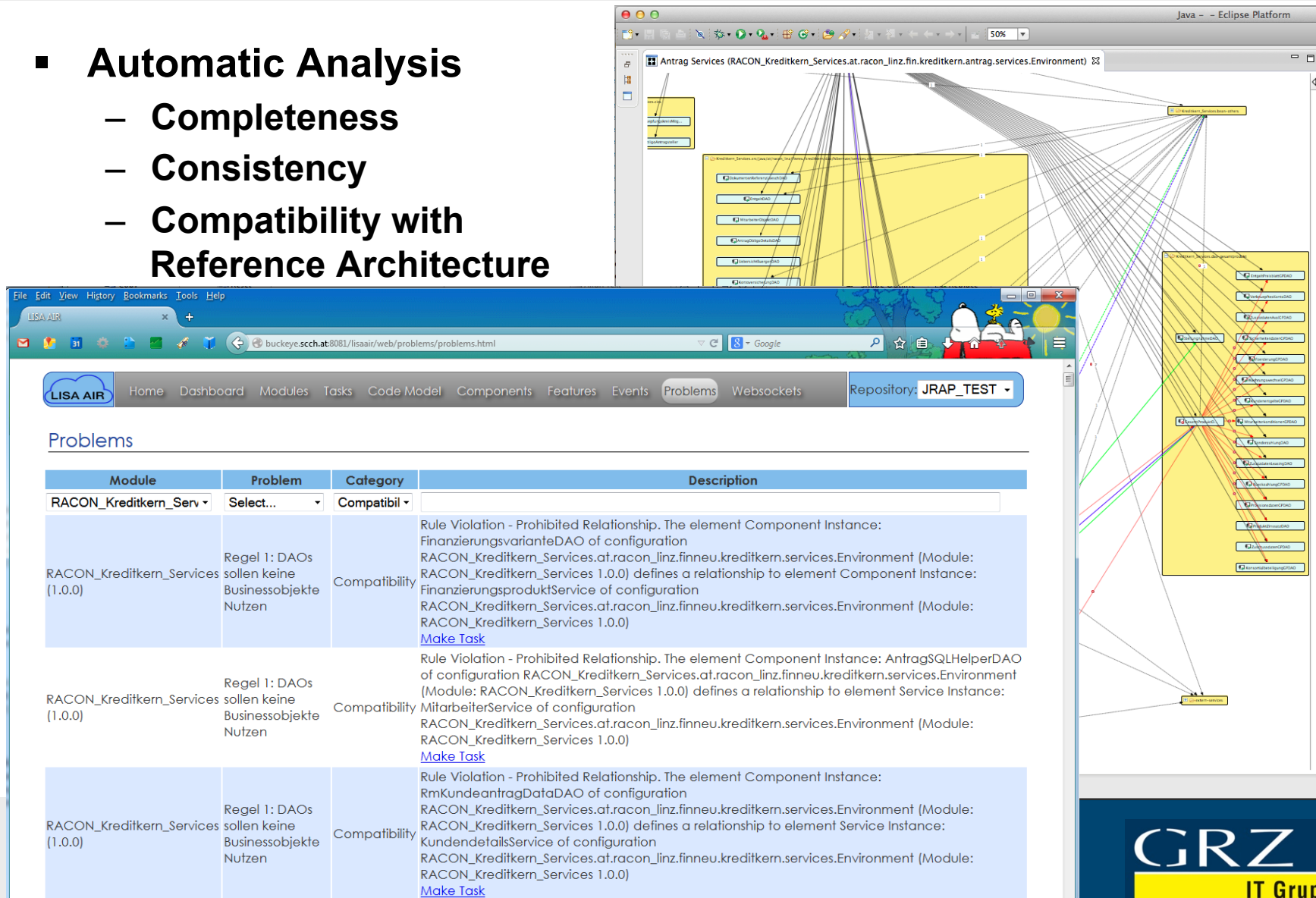
# Architecture Extraction & Visualization

- Up-to-date documentation
- Architectural information in implementation
- Incremental architecture extraction
- Simulated component composition
- **Architectural views**



# Architecture Analysis and Review

- **Automatic Analysis**
  - **Completeness**
  - **Consistency**
  - **Compatibility with Reference Architecture**

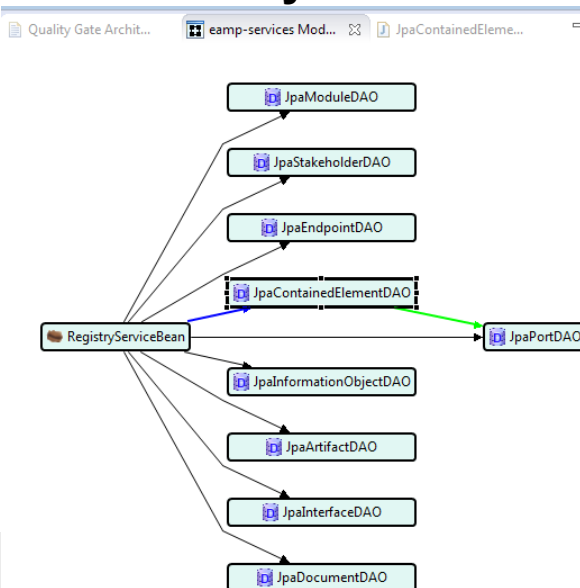


The screenshot shows the LISA AIR web application interface. The top section displays a complex dependency graph with various components and services connected by lines. The bottom section shows a table of problems found during the analysis.

Module	Problem	Category	Description
RACON_Kreditkern_Serv (1.0.0)	Regel 1: DAOs sollen keine Businessobjekte Nutzen	Compatibility	Rule Violation - Prohibited Relationship. The element Component Instance: FinanzierungsvarianteDAO of configuration RACON_Kreditkern_Services.at.racon_linz.finneu.kreditkern.services.Environment (Module: RACON_Kreditkern_Services 1.0.0) defines a relationship to element Component Instance: FinanzierungsproduktService of configuration RACON_Kreditkern_Services.at.racon_linz.finneu.kreditkern.services.Environment (Module: RACON_Kreditkern_Services 1.0.0) <a href="#">Make Task</a>
RACON_Kreditkern_Services (1.0.0)	Regel 1: DAOs sollen keine Businessobjekte Nutzen	Compatibility	Rule Violation - Prohibited Relationship. The element Component Instance: AntragSQLHelperDAO of configuration RACON_Kreditkern_Services.at.racon_linz.finneu.kreditkern.services.Environment (Module: RACON_Kreditkern_Services 1.0.0) defines a relationship to element Service Instance: MitarbeiterService of configuration RACON_Kreditkern_Services.at.racon_linz.finneu.kreditkern.services.Environment (Module: RACON_Kreditkern_Services 1.0.0) <a href="#">Make Task</a>
RACON_Kreditkern_Services (1.0.0)	Regel 1: DAOs sollen keine Businessobjekte Nutzen	Compatibility	Rule Violation - Prohibited Relationship. The element Component Instance: RmKundeAntragDataDAO of configuration RACON_Kreditkern_Services.at.racon_linz.finneu.kreditkern.services.Environment (Module: RACON_Kreditkern_Services 1.0.0) defines a relationship to element Service Instance: KundendetailsService of configuration RACON_Kreditkern_Services.at.racon_linz.finneu.kreditkern.services.Environment (Module: RACON_Kreditkern_Services 1.0.0) <a href="#">Make Task</a>

# Architecture Analysis and Review

- Automatic Analysis
  - Completeness
  - Consistency
  - Compatibility with Reference Architecture
- Manual Analysis
  - Quality Gate Reviews



Quality Gate Architektur | Antrag Services (RACON\_Kreditkern... | eamp-services Module Configuratio... | JpaPortDAO.java

### Assessment der Einhaltung der Architekturrichtlinien

Remarks

- ▷ Geschäftsausrichtung
- ▷ Prozessausrichtung
- ▷ Fachliche Grundanforderungen
- ▷ Sauberer Datenhaushalt
- ▷ Homogene Applikationslandschaft
- ▷ Querschnittsplattformen
- ▲ Technologiestandardisierung
  - ? Unternehmensweite Technologiestandards
  - ? Plattformunabhängigkeit
  - ? Wahrung der Aktualität von Technologien
  - ? Referenzarchitekturen (Konformität Softwarearchitektur)
  - ? Thin-Clients auf Basis Java / Web
- ▷ Applikationsaufbau und Interoperabilität
- ▷ Nonfunktionale Softwareanforderungen
- ▷ Technologiestandardisierung
- ▷ Effiziente Betriebsmechanismen
- ▷ Qualitative Betriebsanforderungen
- ▷ Securitybetrachtung

**Question: Referenzarchitekturen (Konformität Softwarearchitektur)**  
 IT Lösungen werden auf Basis der im Rahmen der Architektur-Governance entwickelten unternehmensspezifischen Referenzarchitekturen entwickelt. Dabei kommen keine als "auslaufend" definierten Technologien zum Einsatz. Applikationen konzipieren / implementieren keinerlei Funktionalität selbst, die in den Standards / Technologien / Frameworks definiert bzw. zur Verfügung gestellt werden.

Answer:  
 DAOs dürfen keine anderen DAOs verwenden!

Severity: (1 - low, 5 -high)  
 3

Need for Action:  
 Must

Status:  
 Open

Elements: 2 (drop elements onto viewer)

- JpaPortDAO
- JpaContainedElementDAO

Outline | Details

- ◆ JRAP Bean Port Binding
- Connections
- ✖ JpaPortDAO

[Open Sequence Diagram](#)

[Related Configuration Diagrams](#)

- eamp-services Module Configuration Diagram

**Review Remarks**

1 Referenzarchitekturen (Konformität Softwarearchitektur) ✖

DAOs dürfen keine anderen DAOs verwenden!

Status: Open Severity: 3 Need for Action: Must

Review: Quality Gate Architektur ✖ 📄

---

▸ Racon Service Module

---

▸ Operations

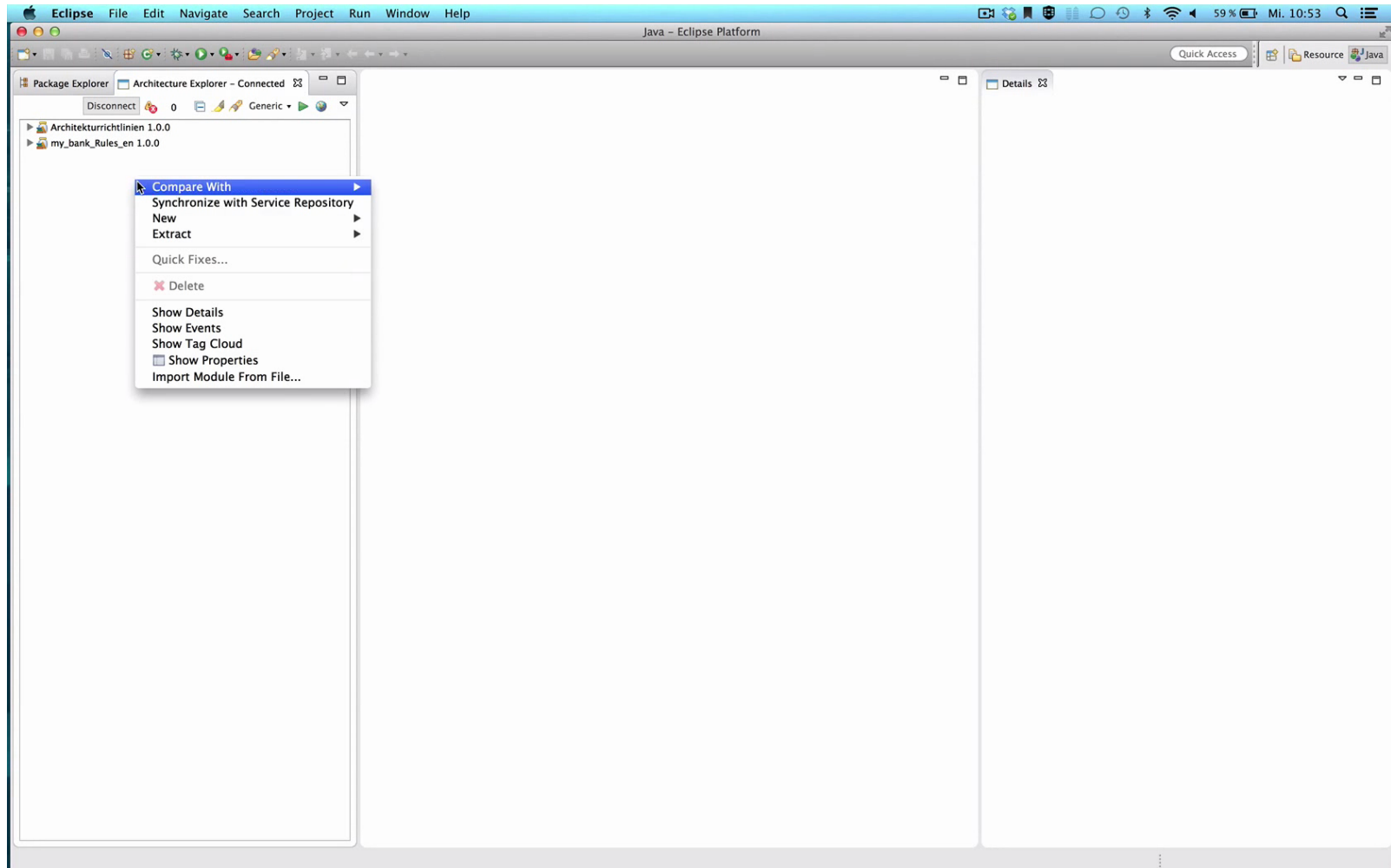
---

▸ Tags (0)

---

▸ Attributes (0)

# Demo





# Lessons Learned

- Model-Driven Development (MDD)
  - + facilitates service development
  - + No boilerplate code, focus on business logic
  - + supports migration to new technology stacks
  
- Model-Based Architecture Management
  - . models need to reflect implementation
  - + supports both automatic and manual architecture analysis
  - + supports governance activities (e.g., repository sync)
  - requires metadata-enhanced implementations (declarative metadata)



# Current and Future Work

- Service Development
  - Investigation of RESTful services
  - Synchronization with Service Registry/Repository
  - Add runtime information to service registry
  
- Architecture Management
  - Better validation of manual review support,
  - Provide global system views (through integration of client and backend systems)
  - Fine tuning (e.g., extraction of publish/subscribe relationships)

# Research Challenges

- Architecture and Testing
  - Facilitate architecture information to identify components and systems that have to be retested based on change impact analysis
- Architecture and Agility
  - Investigate the transition from a rather plan-driven development process to more agile methodologies (developer driven)
  - How to establish agile methodology within the required regulatory requirements and existing organizational structures in the financial domain
- Architecture as a Service
  - Provide architectural information and services to different stakeholders (ongoing work)
- Architecture Knowledge Sharing
  - Develop means to provide architectural information to exactly the organizational units and architects that might be affected by a change.

**Thank you!**

**Questions?**