

JBoss Embedded: A Convergent Architecture for Next-generation Communication Services

Thomas Wenckebach
Systems Engineer, Nokia Siemens Networks
09-02-2009

Agenda

- **NSN Business Support Systems**
 - **Business facts**
 - **Applications and architecture evolution**
- **Convergent Architecture**
 - Some requirements
 - Integration patterns for JBoss@CFRAME
- **JBoss@CFRAME**
 - for Recharge
 - for Call Control
 - JAIN SLEE
 - plain JEE
 - Convergent Service Creation Environment

NSN Business Support Systems

- *Mission and experience*

Our mission:

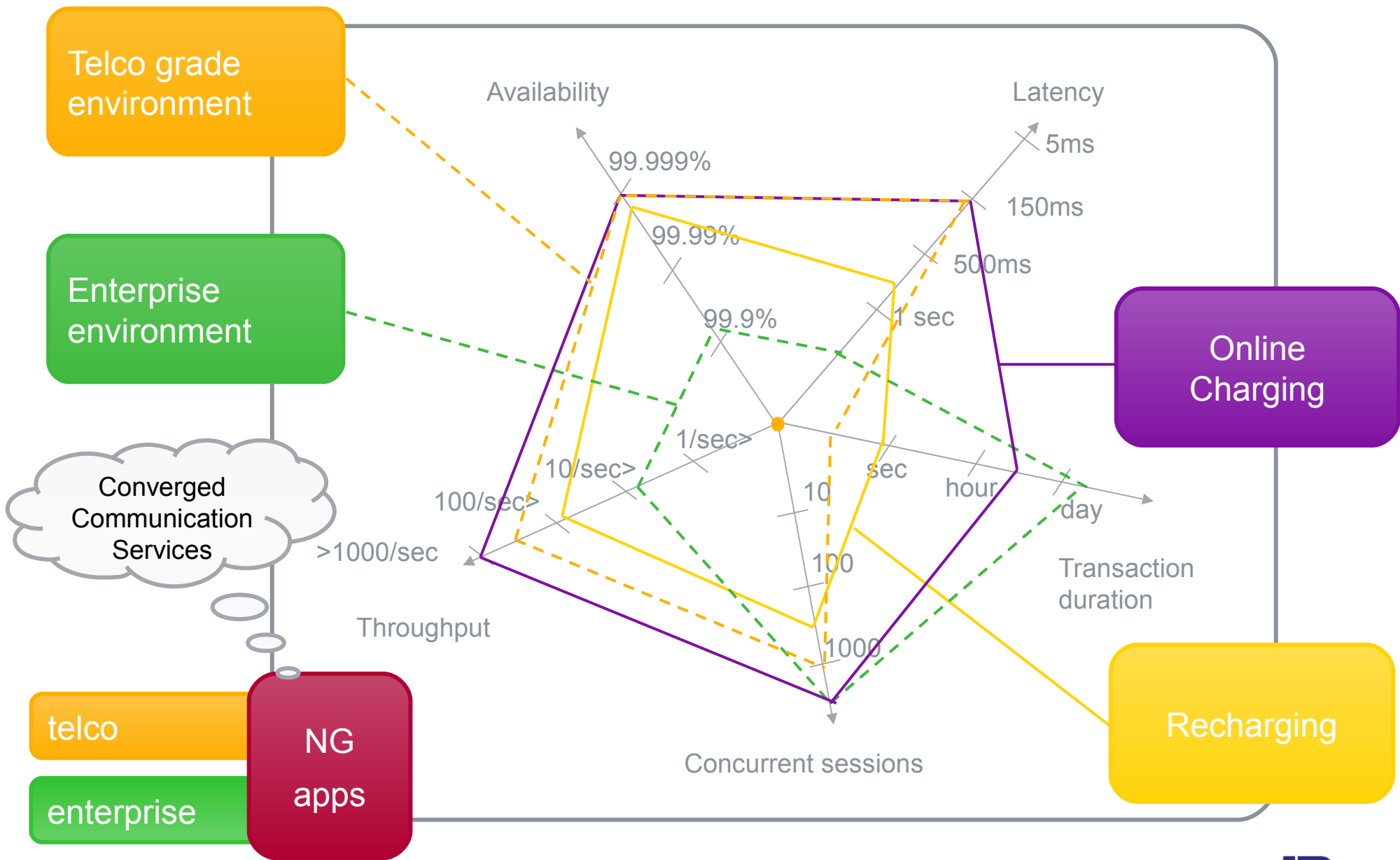
Monetize the value of connecting communities

Our charging experience

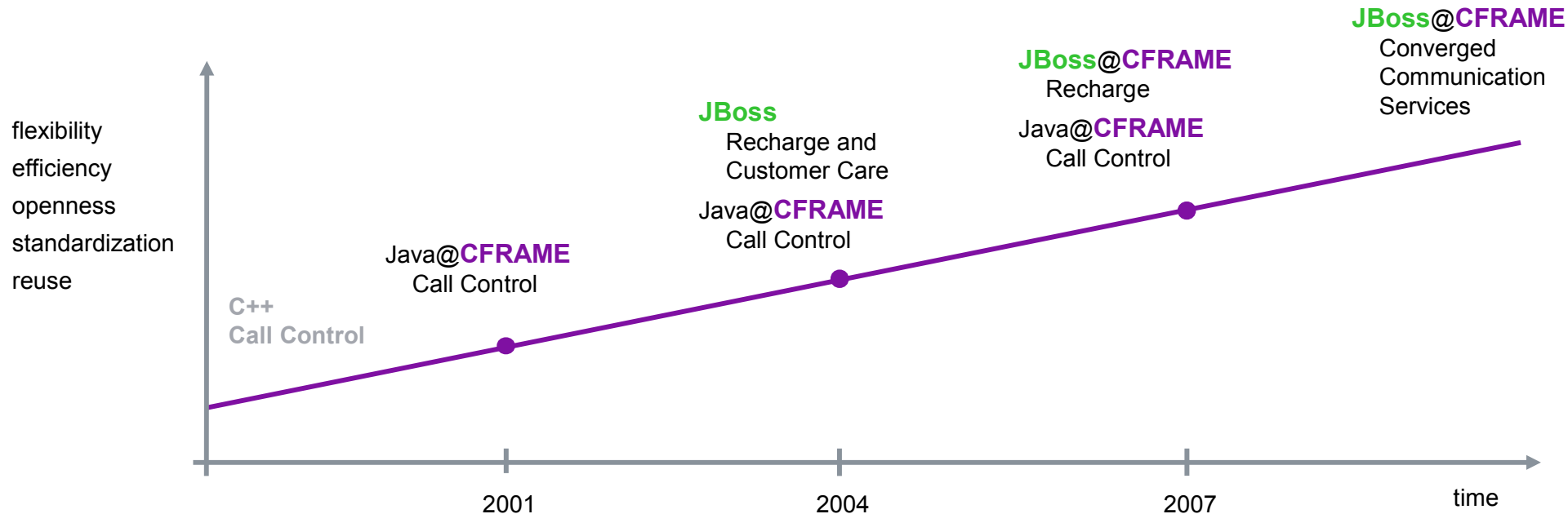
- World #1 for online charging solutions
- **20%** market share
- Serving more than **500** million online customers
- More than **250** charging customers
- Over **500** systems integration projects globally
- First generation convergent charging solution in commercial use



Quality attributes define application classes



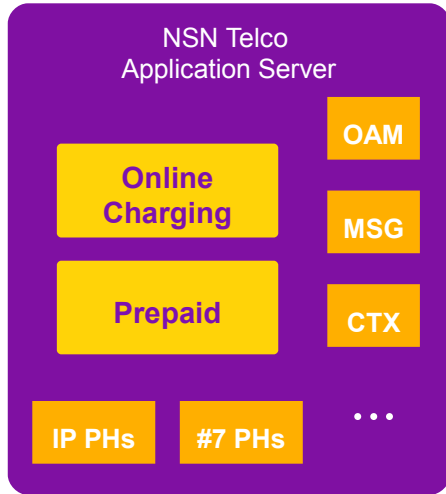
Convergent Architecture evolution



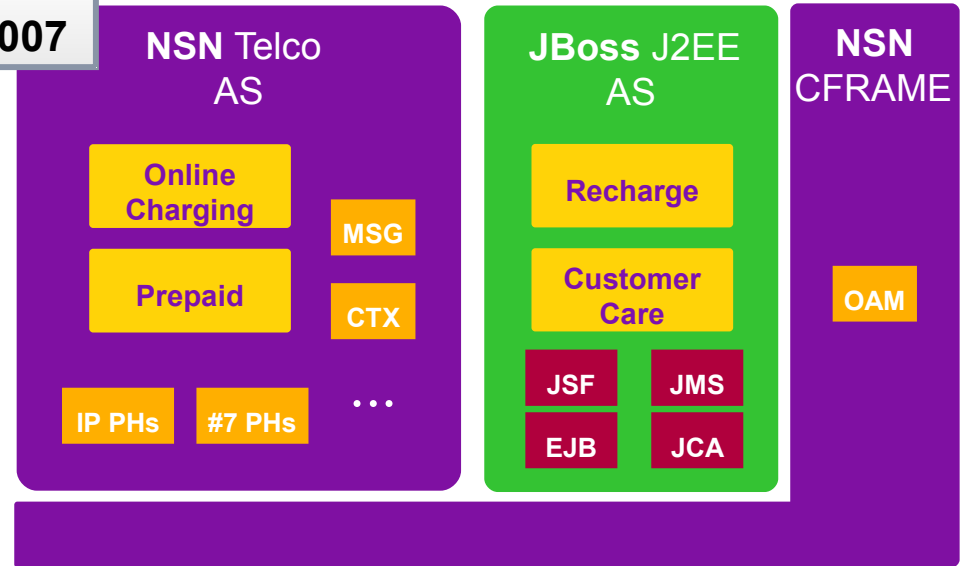
- Use proven assets from both worlds: Telco (C++/ Java) and Enterprise (Java)
 - NSN telco framework CFRAME provides features and facilities superior to JEE servers
 - using open JEE environments increases efficiency and flexibility in solution development
- Telco services (IN) --> converged communication services (NGIN, Internet, IT)
 - support **Java Enterprise** style application development
 - open IFs and programming paradigms (in-house or by customer!)
 - **BUT:** fulfill demanding **carrier-grade** requirements: reliability, throughput, availability, scalability
 - support usage of COTS IT components built using standard API
- Evolution for NSN charging and care solutions

Convergent Architecture Evolution: Overview

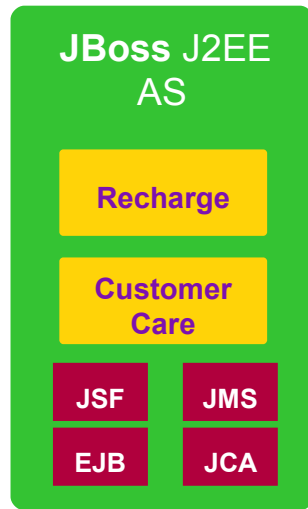
2001



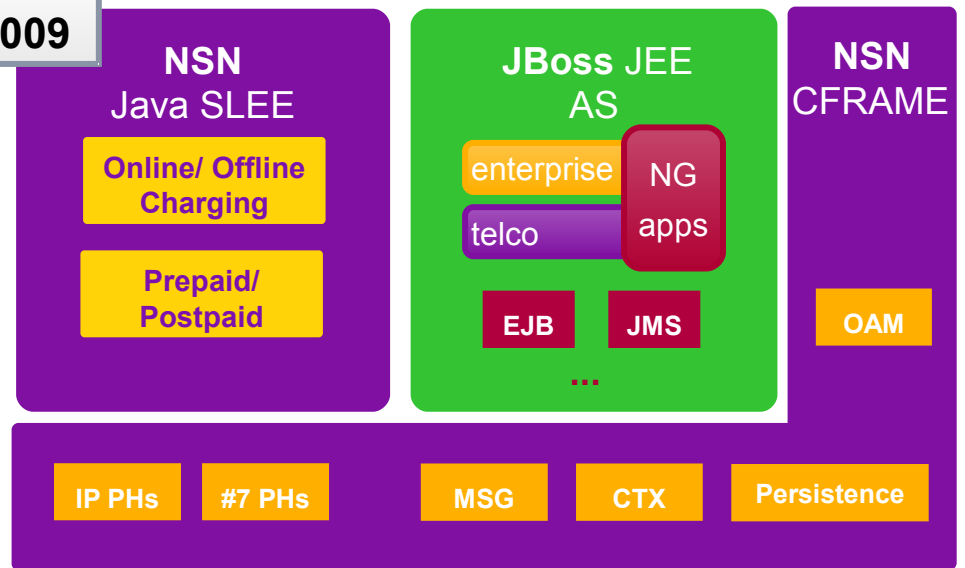
2007



2004



2009



Agenda

- NSN Business Support Systems
 - Business facts
 - Applications and architectures
- **Convergent Architecture**
 - **Some requirements**
 - **Integration patterns for JBoss@CFRAME**
- JBoss@CFRAME
 - for Recharge
 - for Call Control
 - JAIN SLEE
 - plain JEE
 - Convergent Service Creation Environment

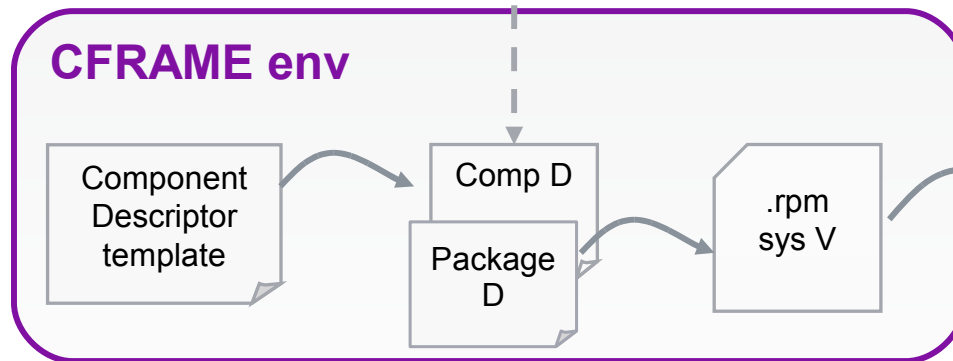
Convergent Architecture - some requirements

(I) Development Process

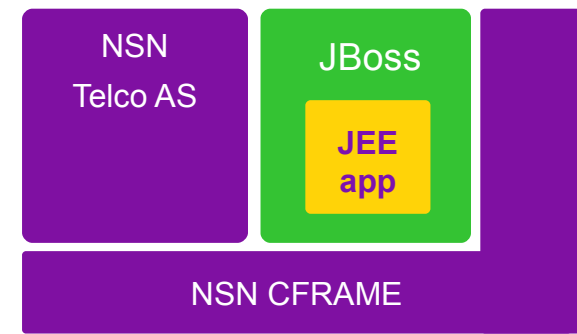
- Make available telco-grade Platform Services for Java Enterprise modules
 - “transparent” for application: stay in JEE development environment
 - without changing the source code of the 3rd-party AS



1. **Develop** and **test** application in open JEE environment
2. Install via CFRAME Component Descriptor



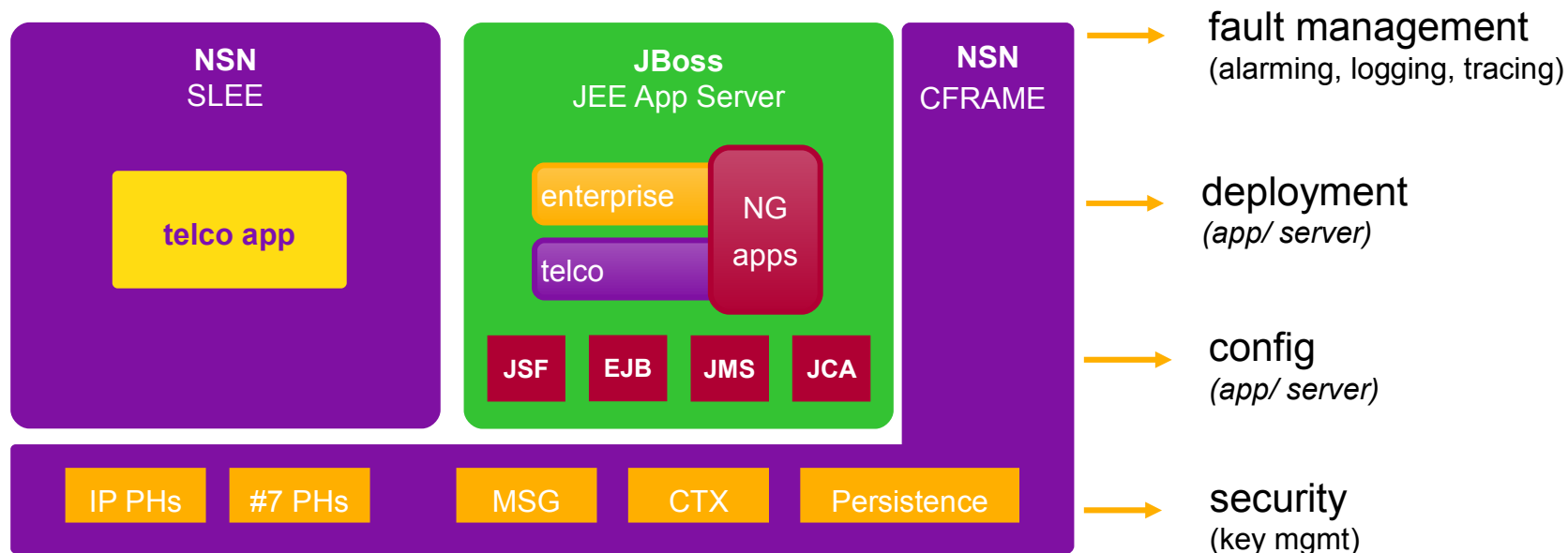
Runtime



Convergent Architecture - some requirements

(II) Business requirements

- Keep homogeneous telco-grade OAM framework
 - fault mgmt, installation, configuration, etc.
 - avoid deployment of inhomogeneous servers



- Keep telco-grade resource adaptors and platform services
 - IP and #7 Protocol Handlers
 - Messaging, Contexts (session data), Service Data Persistence (application data)

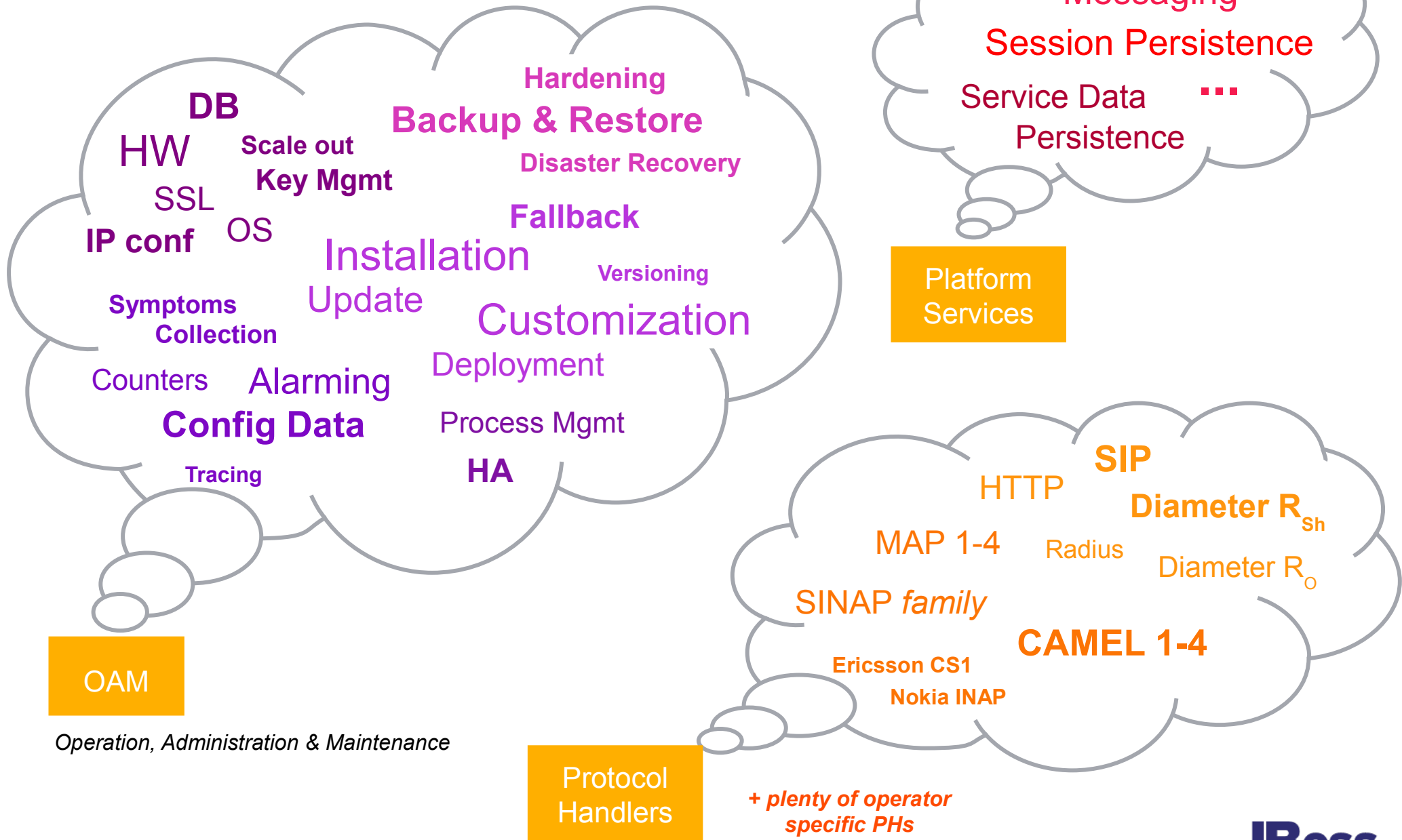
Convergent Architecture - some requirements

(III) Derived requirements

- CFFRAME must keep control over processes, in particular JVM
 - own CFFRAME socket factory must be set in JVM programmatically
 - telco platform services must be able to run in separate threads (native, Java)

- 3rd party AS must be able
 - to run in **embedded mode**, such that JBoss is instantiated within same address space as efficient telco-grade NSN platform services
 - highly **modular, configurable, pluggable, extendible**

Proposed value-add JEE ↔ CFRAME



OAM

Operation, Administration & Maintenance

Platform Services

Protocol Handlers

+ plenty of operator specific PHs



Foundation: JBoss Embedded

Integration Patterns

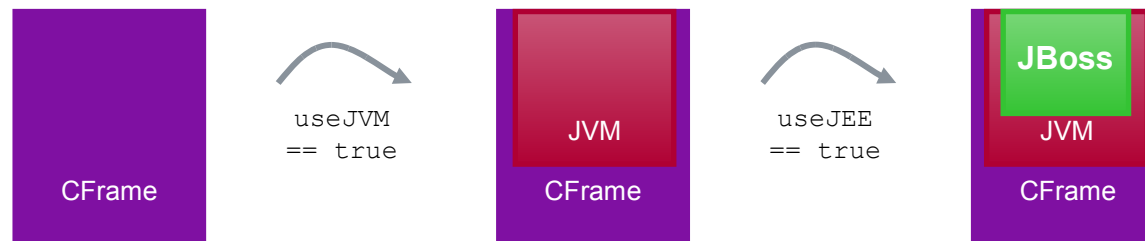
- Templating
- Service Provider Framework/ Interface
- Modularization
- Aspect Oriented Programming
- Façade

Integrated CFRAME functions and used patterns:

server configuration: **Templating**, two-stage parameterized file IF (*JBoss*)
server customization: **Modularization**, service modularization via JMX Microkernel (*JBoss*)
clustering: JBoss **Embedded**, self-organizing cluster membership (*JGroups*)
application configuration: **SPI**, MBean Persister (*JBoss*)
security and key management: **SPI**, Java socket factory for SSL(*JDK*)
tracing: **SPI**, logging appender (*log4j*)
alarming: **SPI**, logging appender (*log4j*)
contexts: **SPI**, Persistence Provider (*JPA/JEE*)
thread supervision: **AOP**, notify supervisor before/ after invocations within interceptors (*JBoss*)
licensing: **AOP**, check with license manager before/ after invocations within interceptors (*JBoss*)
alarming: **Façade**, EJB as façade to NSN platform specific alarming framework (*Java*)
performance monitoring: **Façade**, EJB as façade to NSN platform specific performance counters (*Java*)

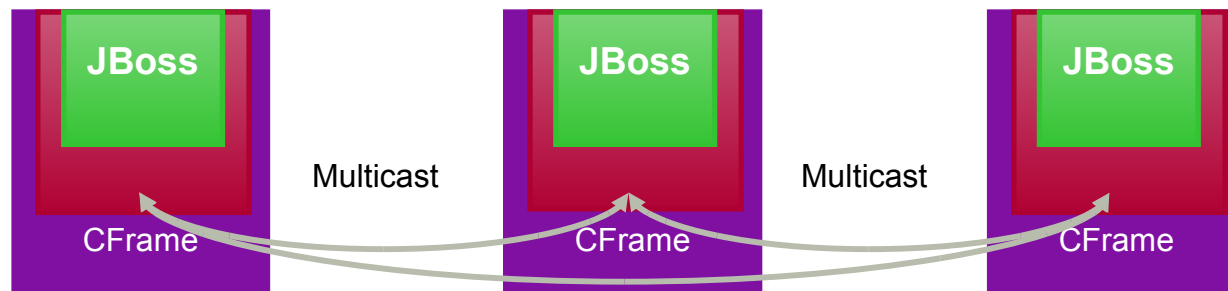
Foundation: JBoss Embedded

- CFRAME starts JBoss within existing JVM by creating a separate thread and invoking the `static main` method of the `org.jboss.Main` class



Coexistence of JBoss Clustering and CFRAME Process Groups w/o interference

- JBoss Clustering: self-organizing cluster membership via group communication (JGroups)
 - is able to dynamically form clusters (= process groups) via IP multicast or set of TCP connections
 - dynamically assigns group master: ***no external master process!***
- CFRAME starts JBoss as one or several process group(s) with configurable number of instances on a configurable number of nodes (*process based scaling*)

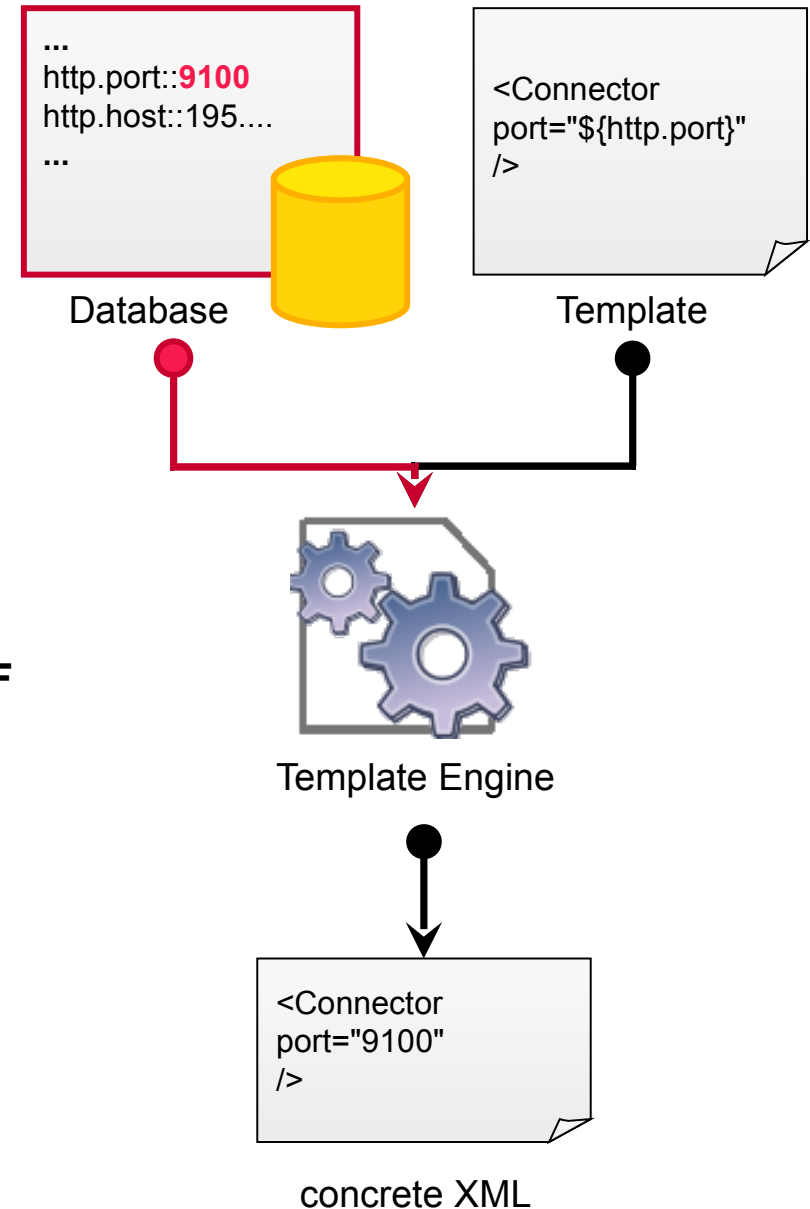


Pattern Templating:

- Template files containing placeholders or modifiable content to be resolved and processed by template engine
- Separation of static structure (with default content) and runtime context specific content

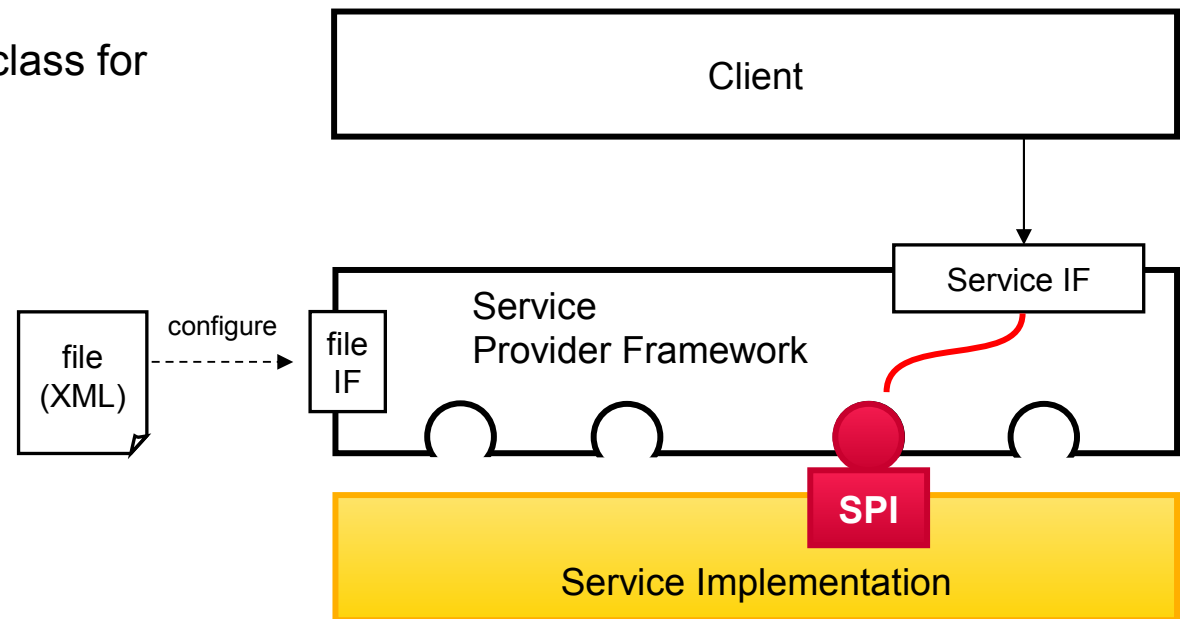
Templating example:

- JBoss service configuration: JBoss XMBean is an XML **file IF** for **static** configuration data
- define attributes as plain text or via `${java.sys.prop}`
 - dynamically resolved by JBoss `StringPropertyReplacer`
 - more powerful: JBoss Binding Service for XSLT support
- CFRAME sets system properties, backed by own DB-based configuration framework:
 - management of config data according to lifecycle classes



Pattern Service Provider Framework:

- Declarative/ programmatical def. of impl class for Service Provider Interface (SPI)
- Application is
 - compiled against known (open) IFs
 - linked against implementation using existing telco framework



Service Provider Framework example:

- JBoss MBean persistence for **dynamic** configuration data
- Integrated into CFRAME's DB-based configuration framework: own impl of SPI
`org.jboss.mx.persistence.PersistenceManager`
 - definition of impl class via XML file: XMLElement definition in `jboss-service.xml` files
 - load on creation (XMLElement initialization):

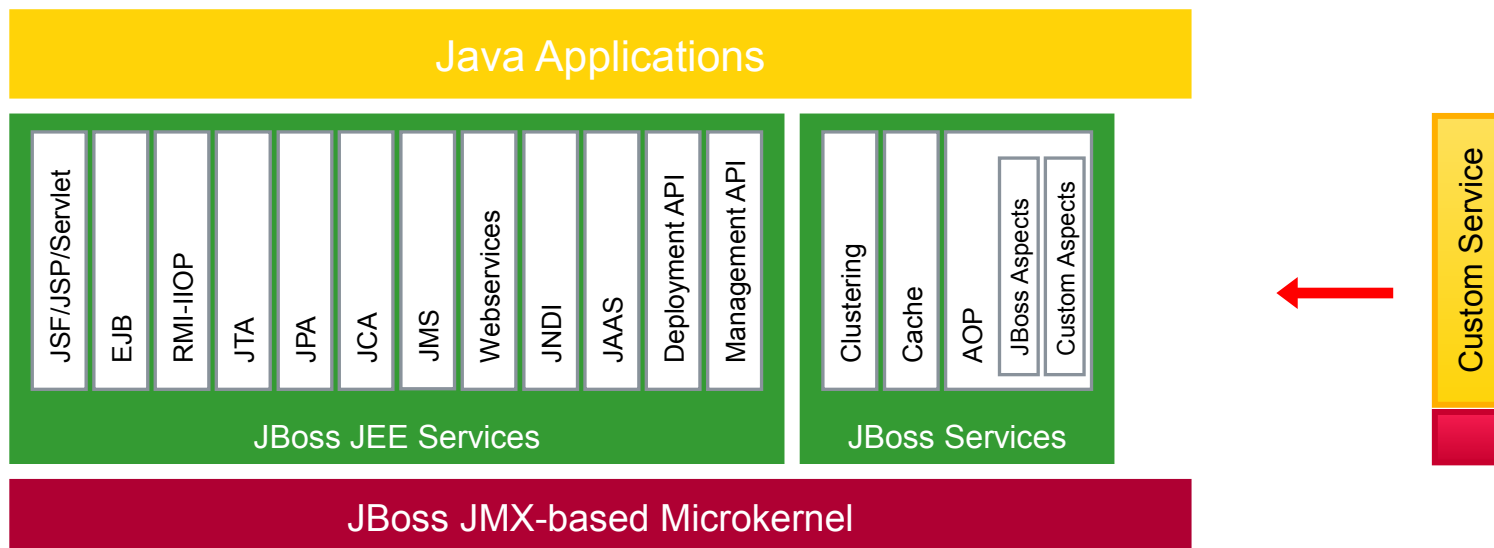
```
persistence = (PersistenceManager) server.instantiate("persistence-manager");
persistence.load(...);
```
 - store during runtime:

```
XMLElement.setAttribute(Attribute attribute) --> persistence.store(...)
```
- Also direction CFRAME DB --> JBoss provided via own updater JBoss service

JBoss Integration with CFRAME (cont.)

Pattern Modularization:

- "Non-distributed SOA": a service is autonomous in nature w.r.t. *implementation, life-cycle, (location, platform)*
- JMX Microkernel: pioneer approach based on extensions of JMX, imitated by recent architectures based on OSGi (Oracle Weblogic, IBM Websphere, GlassFish, Spring dm)

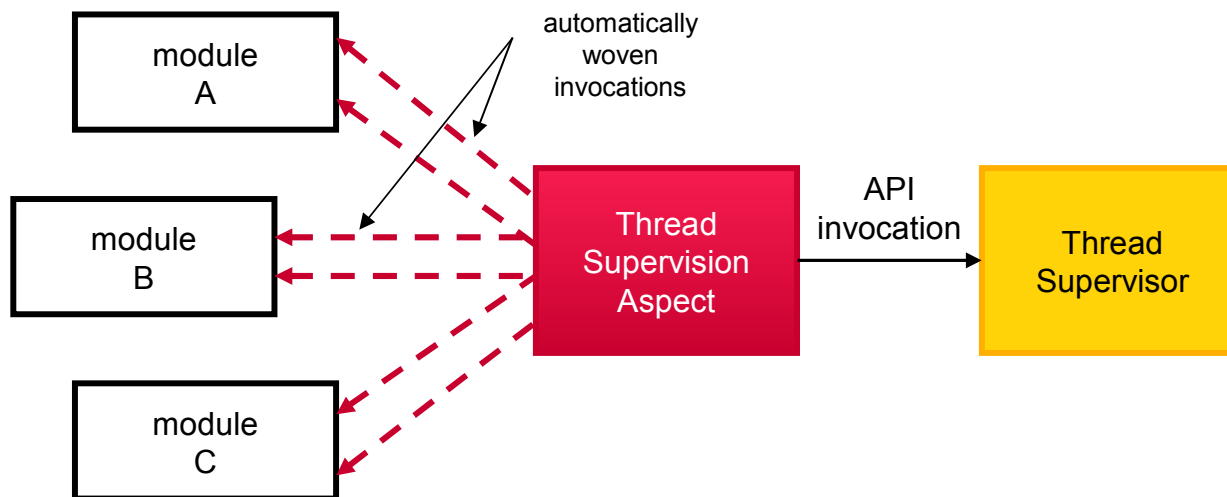


Modularization example:

- Application server customization: JBoss AS is designed service-oriented
 - toolbox for customization, services are plugged into the underlying JMX micro kernel as XMBEans
 - allows for easy modification or extension of the JBoss service set by new JBoss services
- CFRAME XMBEAN updater service as companion to own MBean Persister

Pattern **Aspect Oriented Programming (AOP)**:

- Good for separation of concerns, but also a great integration technique!
- Definition of pointcuts via Java annotations, plain XML entries (e.g. JBoss container interceptors) or meta language based expressions (e.g. JBoss AOP)
- Execution of advice via callbacks or runtime/load-time/compile-time byte-code weaving

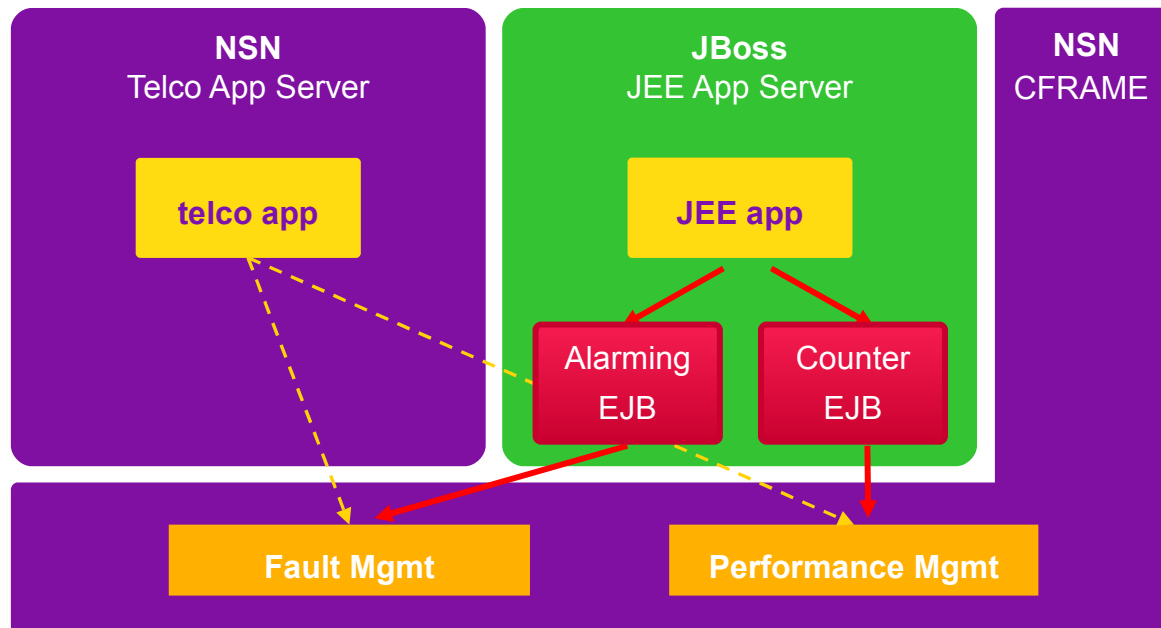


Aspect Oriented Programming example:

- Inject platform code on enter/leave request processing methods
- JBoss interceptor for thread supervision, i.e. detect hanging resources or endless loops
- Add to interceptor chain for respective invokers in `standardjboss.xml`

Pattern **EJB Façade**:

- EJB façades forward service requests to telco platform via library calls, RPC or messaging
- Support of known (open, de facto standard) programming model
--> app is programmed against proprietary business IF, but the overall programming paradigm is adapted to the Java Enterprise style



EJB Façade example:

- Alarming: EJB as façade to NSN Fault Mgmt framework

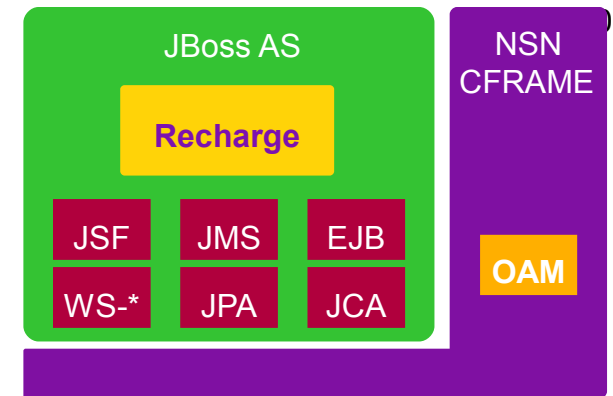
Agenda

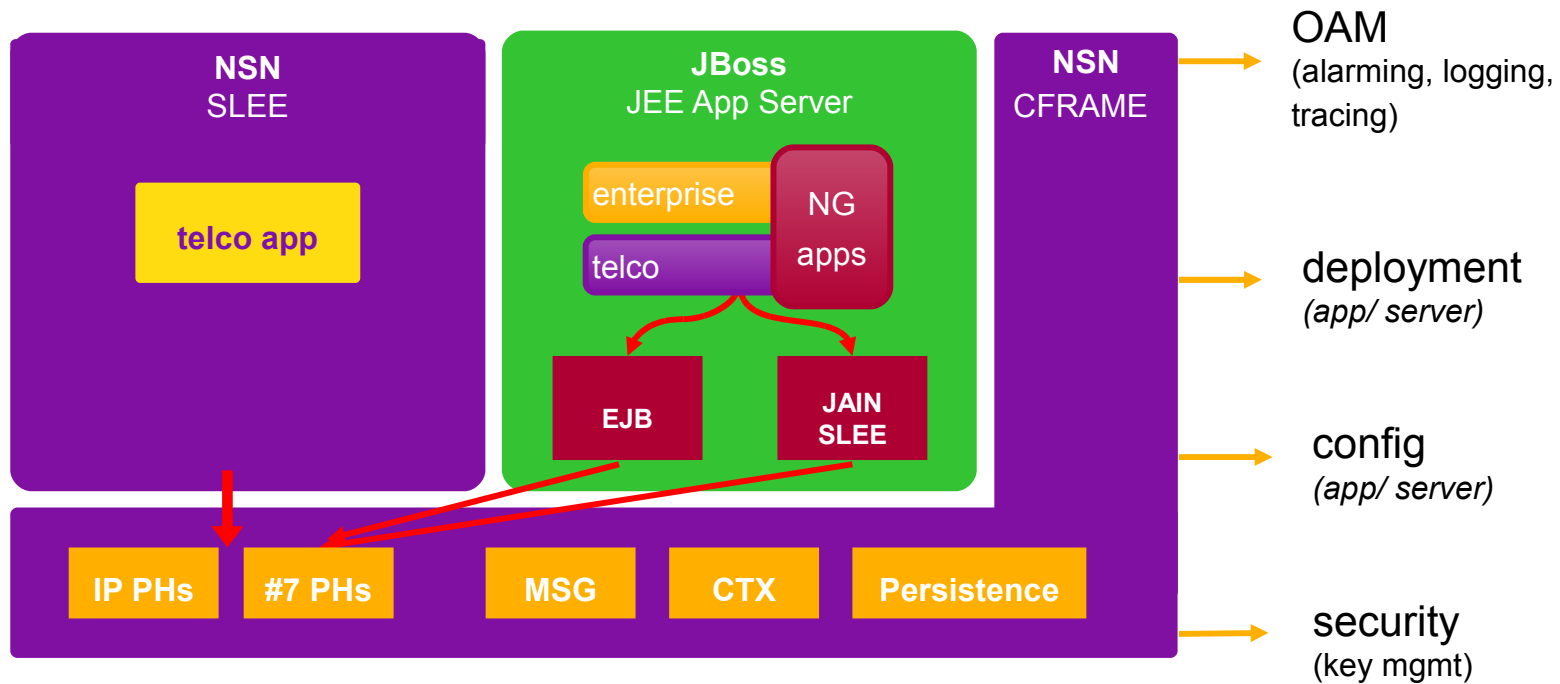
- NSN Business Support Systems
 - Business facts
 - Applications and architectures
- Convergent Architecture
 - Some requirements
 - Integration patterns for JBoss@CFRAME
- **JBoss@CFRAME**
 - **for Recharge**
 - **for Call Control**
 - **JAIN SLEE**
 - **plain JEE**
 - **Convergent Service Creation Environment**

Recharge: charge@once topup 5.0

– a high-end enterprise application

- charge@once topup 5.0 solution offers
 - Top up (recharge prepaid card), Mobile payment
 - Voucher, Reseller management
- Supports self care channels (SMS, USSD, IVR, Web) and customer care or resellers access, **all in real time**
- Payment via vouchers, cash or via online debiting customers & requestor credit/debit card or bank account as well as by transferring money from one account/balance to another account/balance.
- Strong point: *system scalability* and *performance* characteristics, allowing e.g. for mass micro-topup scenarios powered by a low-cost infra-structure
- **JBoss 4.2.3GA** without changes to source code
 - JBoss Microkernel with MBean persistence
 - JSF with JBoss SEAM
 - EJB 3.0
 - JMS/ MDBs
 - JPA/ Hibernate with Oracle 11 RAC
 - Webservices, JCA for enterprise application integration (back-end systems)

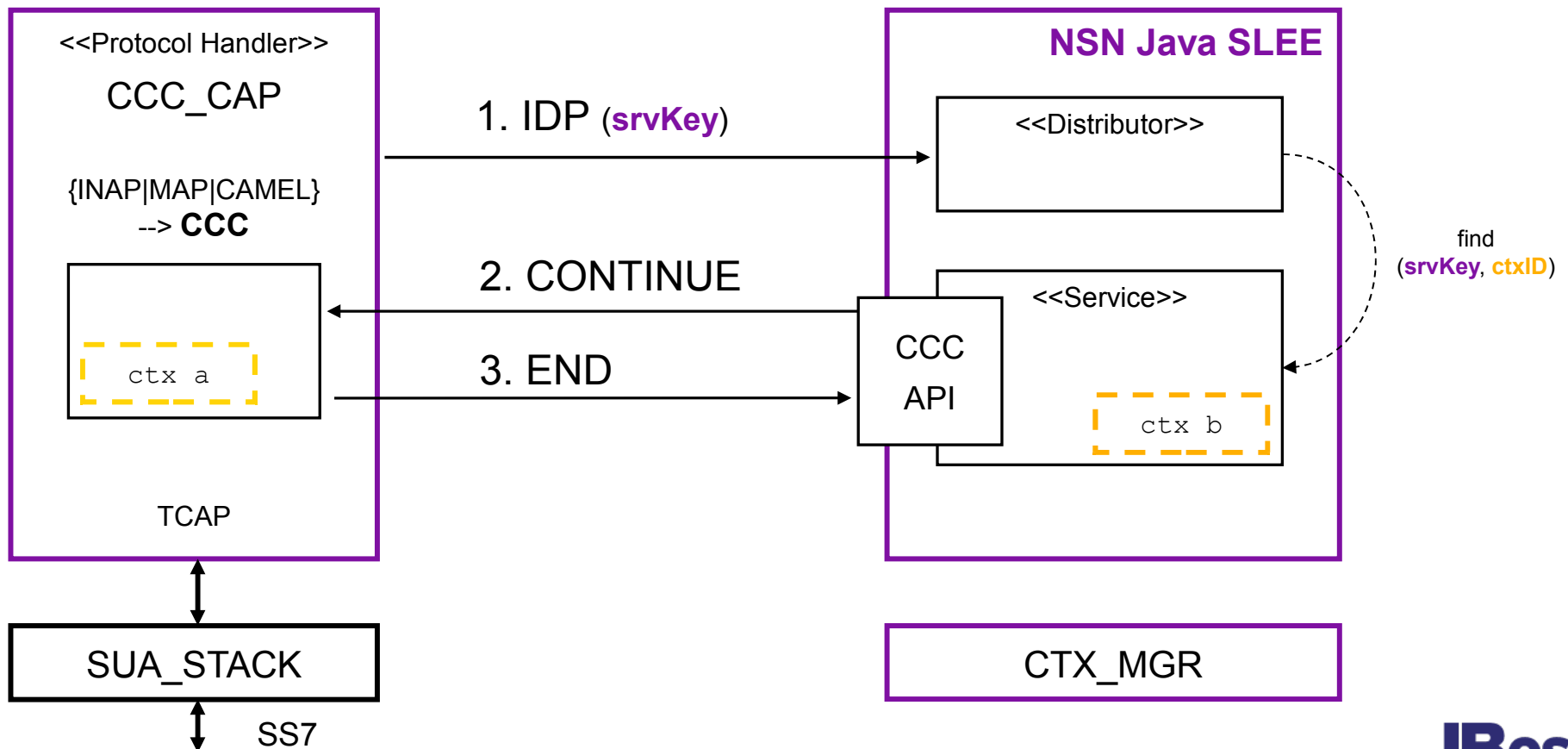




- Make available telco-grade Platform Services for JEE applications
--> Messaging, Session Persistence (CTX), Service Data Persistence
- Communication services on JBoss, using existing NSN protocol handlers (PH)
- *Two approaches:*
 - use JAIN SLEE container on top of JBoss, encapsulate Protocol Handlers as Resource Adaptors
 - use plain JEE environment

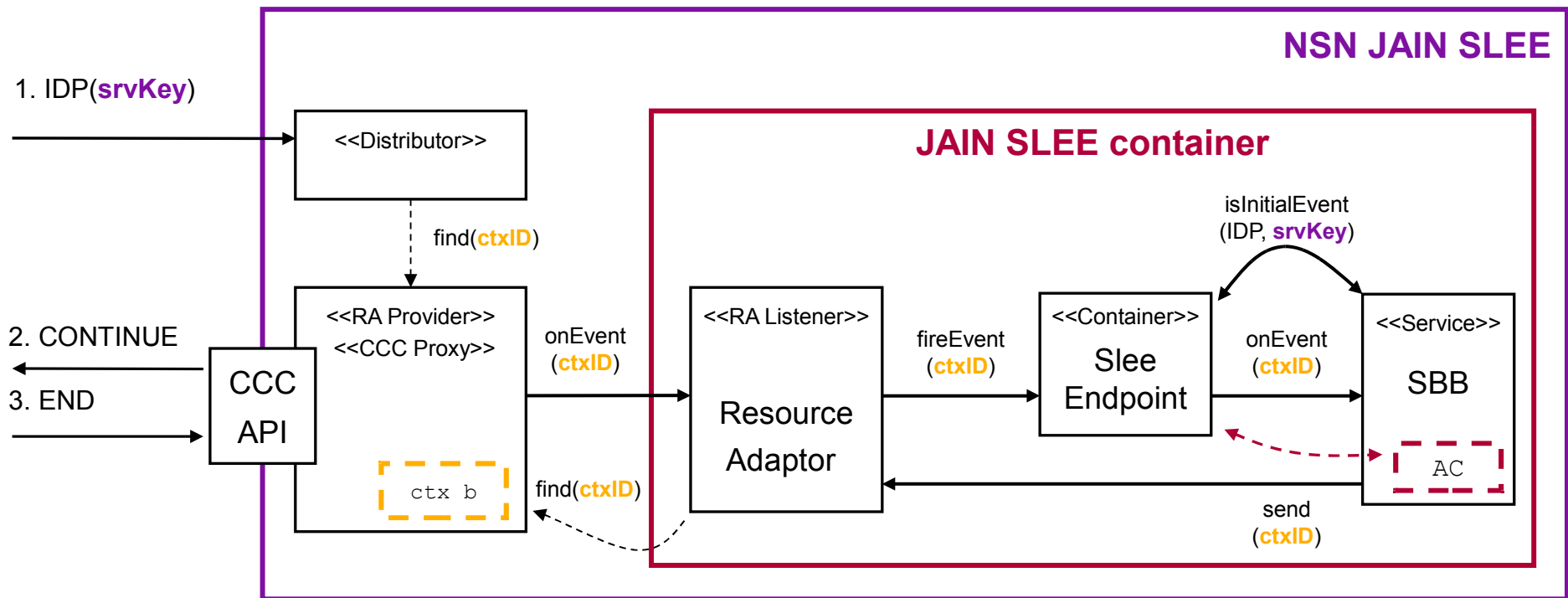
Common Call Control on Java SLEE

- Protocol abstraction via NSN **Common Call Control API** – much richer than **JAIN JCC!**
- Components communicate via messages: *event-driven architecture*
- Trigger table for mapping: initial event (e.g. IDP) + trigger attributes (**srvKey**,...) --> Service bean
- Follow-up conversation is done Context specific (ctx = session/ dialog state)
 - ctxIDs are part of address (*consistent hashing*)
 - protocol handler "CCC_CAP" and NSN Java SLEE maintain dialog (identified via **ctx a:ctx b**)
 - but: Service bean is stateless (pool!): state is stored in ctx, which injected into bean and replicated (async)



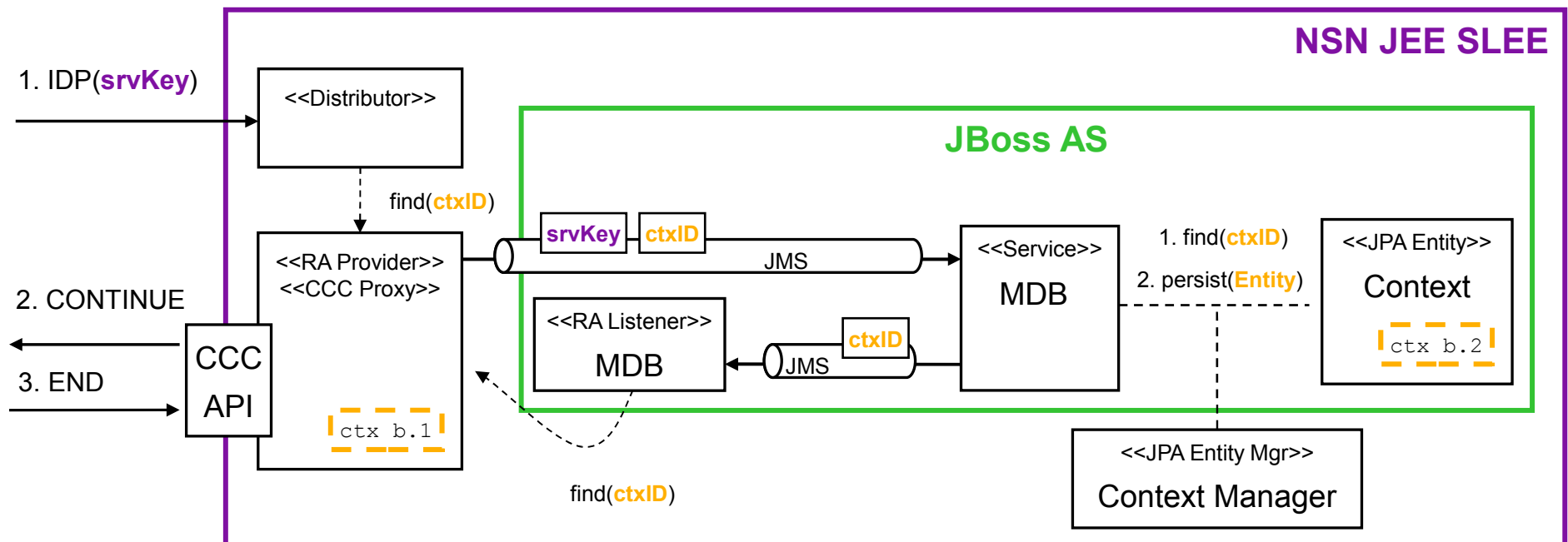
Common Call Control on **JAIN SLEE**

- Most relevant concepts in NSN Java SLEE already addressed by JSLEE 1.1 standard (JSR 240):
 - SBBs, Events (incl. service triggering), Activity Context (AC, cf. JBossCache), Resource Adaptors (RA), Persistence (e.g. via JPA/ Hibernate)
- NSN **ctxID** is encoded in ActivityHandle; Service has separate context **AC** (provided by SLEE)
- No support for initial event selection via **srvKey**
 - SBB via custom callback method ("initial-event-selector-method") `isInitialEvent()`



Common Call Control on plain JEE

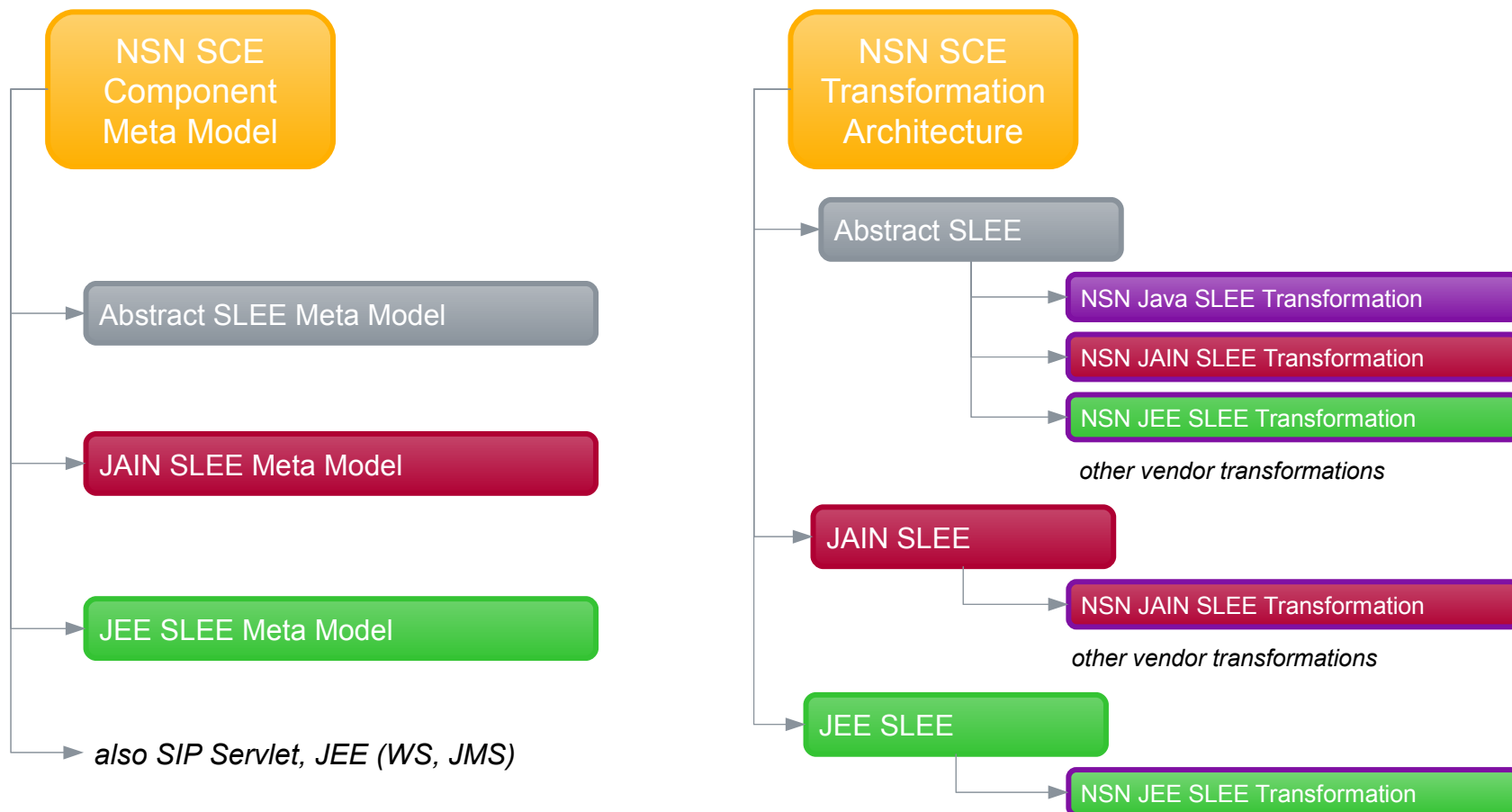
- Service realized as MDB with request/response queues per "Resource Adaptor"
 - no decentralized event subscription as in JSLEE: operator has to ensure *service isolation*
- CCC API mapped to JMS ObjectMessage(s)
- Service triggering: JMS Message Selector based on custom **srvKey** Property
- Event sequencing: make sure msgs with same **ctxID** are handled by same MDB instance!
 - JMS 1.1: "All providers must support [...] **JMSXGroupID** and **JMSXGroupSeq**"
- Context (session state) via JPA: interpretation as session store (cf. JBoss Infinispan "POJO Caching"); JPA persistence provider from CFRAME
- **ctxID** for sequencing (JMSXGroupID) and correlation (JMSCorrelationID)



NSN Convergent Service Creation Environment

- Model Driven Software Development

- Service logic is coded in Java (with IDE plugin), or *modelled* with Groovy or Service Designer GUI:
 - Meta Model defines the form; either abstract or oriented towards target environment
 - Transformation implements the form for a target system (XML, if necessary Java, ...)
 - application developer ,fills in‘ the form



Thank you!

thomas.wenckebach@nsn.com

Questions & Answers

QUESTIONS?

**TELL US WHAT YOU THINK:
[REDHAT.COM/JBOSSWORLD-SURVEY](https://redhat.com/jboss-world-survey)**