OpenIDM

Yesterday, Today and Tomorrow

Radovan Semančík

January 2011

Agenda

- OpenIDM Yesterday
- OpenIDM Today
- OpenIDM Tomorrow

nLight

ForgeRock

But before we get to that ...

What OpenIDM Should Be?

What is OpenIDM

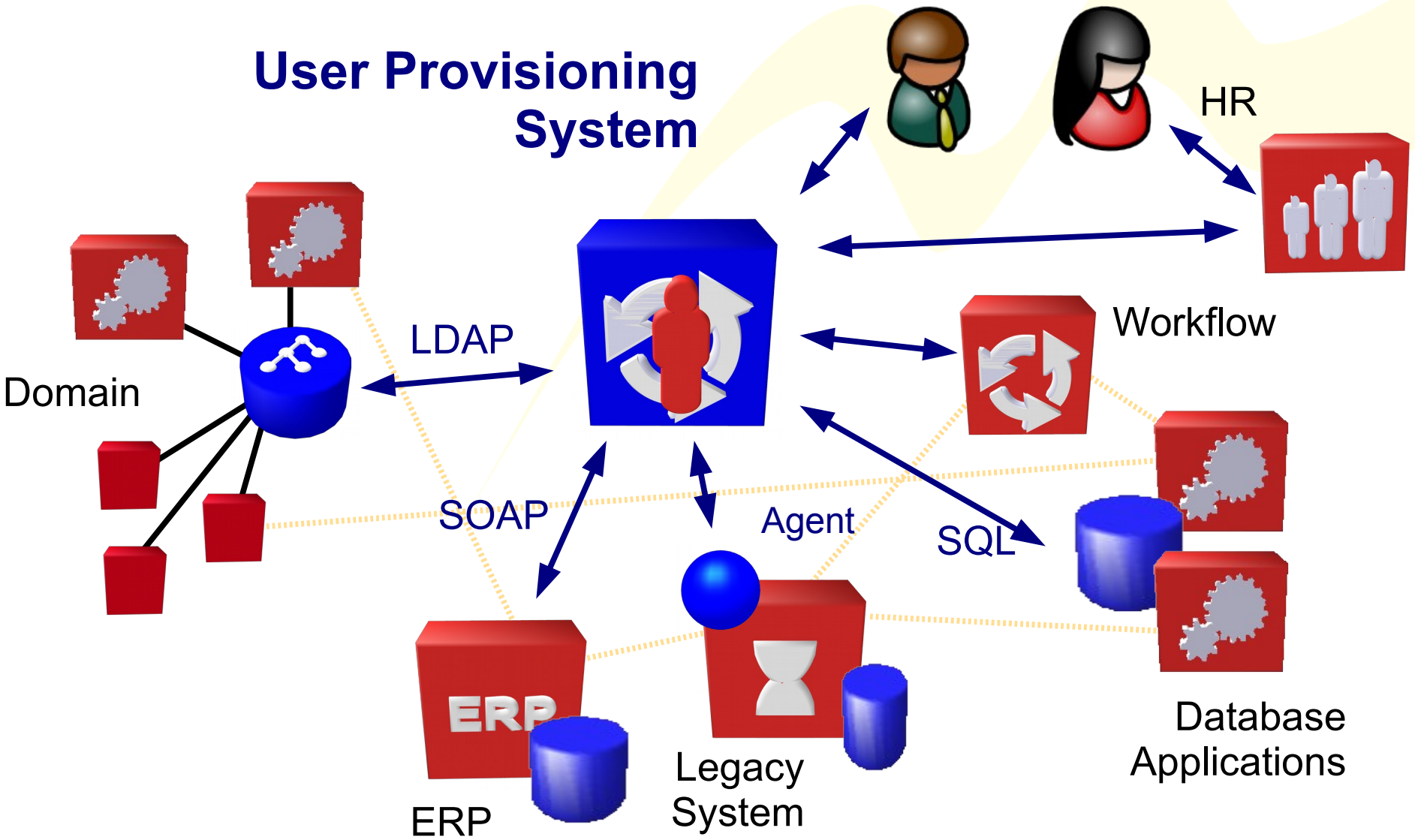
- **Open source provisioning system**
  - Identity Lifecycle Management & Integration
  - Fully open-source system (CDDL)
  - Brand new system, modern architecture
- **Ready for commercial Operation**
  - Reliable, Maintainable, Flexible

ForgeRock

OpenIDM Goals

- ***Provisioning* system**
  - Provisioning, de-provisioning, re-provisioning
  - Data synchronization (automation)
  - Enforce access policies (e.g. RBAC)
  - Support manual activities and exceptions
- **Goal: Identity data consolidation, cleanup and maintenance**

Integration, Integration, Integration!



OpenIDM Goals

- ***Next Generation* system**
  - Component-based system (ESB, OSGi, modular)
  - High-level languages (BPEL, XPath, XSLT, ...)
  - Integration (Web Sevices)
  - … and other buzzwords ...
- **Goal: <u>Efficiently</u> extensible and customizable**

OpenIDM Goals

- ***Provisioning system construction kit***
  - LEGO-like set of components
  - Can be re-combined if needed
  - Choose components that you need
  - Replace some components with others
  - But still have a “default” structure for 80% of cases
- **Goal: Flexibility, flexibility, flexibility**

OpenIDM Goals

- ***Usable* system**
  - Development environment (Netbeans)
  - Fast development cycles (develop-test-deploy)
  - Diagnostics and debugging
  - Open and documented source
  - Open development process
- **Goal: Deploy and maintain with reasonable cost**

Basic Principles

- **Do not re-invent the wheel**
  - Use industry standards and proven mechanisms if possible
    - e.g. ESB/JBI, BPEL, XPath, JSF, …
  - But avoid over-complicating the system
    - e.g. SPML

Basic Principles

- **XML Everywhere**
  - Data in XML: strict (yet extensible) schemas
  - Interfaces in XML: WSDL
  - Schemas in XML: XSD
    - XSD Schemas for everything (including resources)
  - Efficiency: XML infoset instead of string XML
    - If needed, avoid premature optimizations

Easy Development

- **Detect bugs as early as possible**
  - XML schemas to validate data and configuration
  - Unit tests and validators
- **Architecture-driven, but still considerably agile**
  - Architecture is a guideline, not a dogma
  - Engineering feedback is essential (prototypes, PoC)
  - Customer feedback is the most precious information

OpenIDM Yesterday

or How We Have Originally Designed It

Abstract Architecture

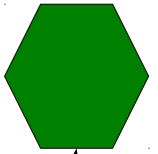
Administrators
Users

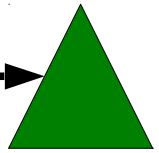
User Interface

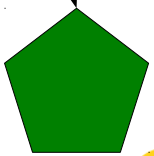
Business Logic

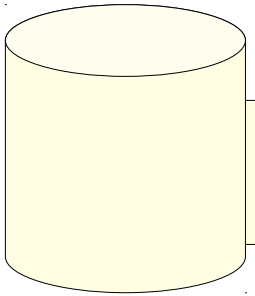
IDM Model

Resources









Repository

Provisioning

ForgeRock

nLight

Abstract Architecture

Administrators
Users

User Interface

Business Logic

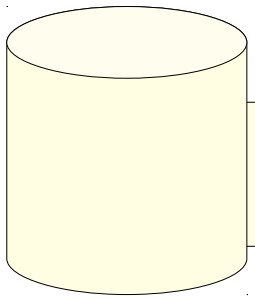
Processing the policies

IDM Model

RBAC,
ABAC,
Hybrid, ...

Resources

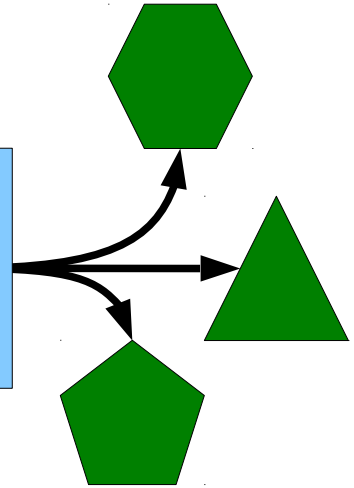
Relational Database
(or maybe LDAP later)



Repository

Provisioning

IC

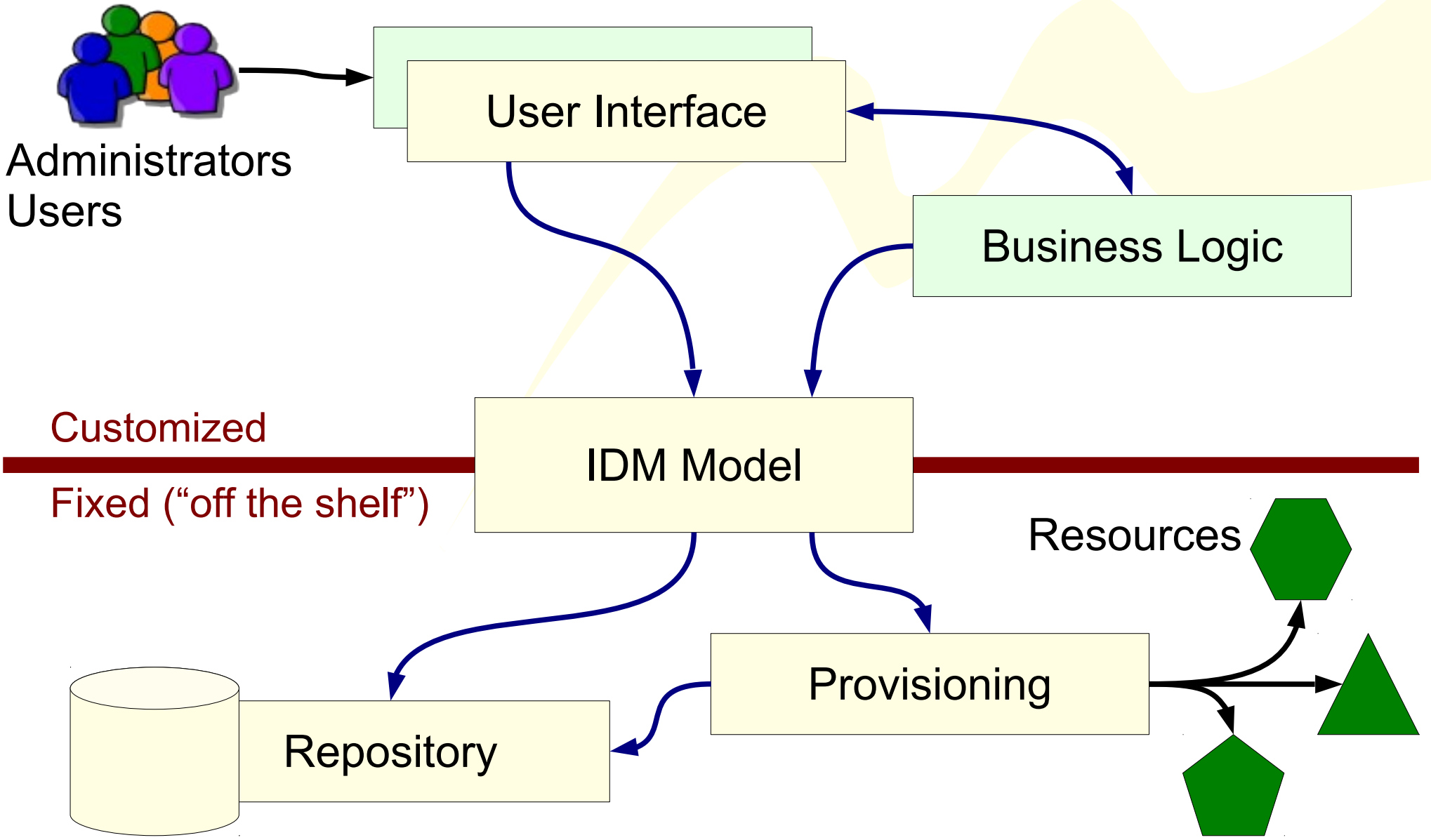


Identity Connector
Framework

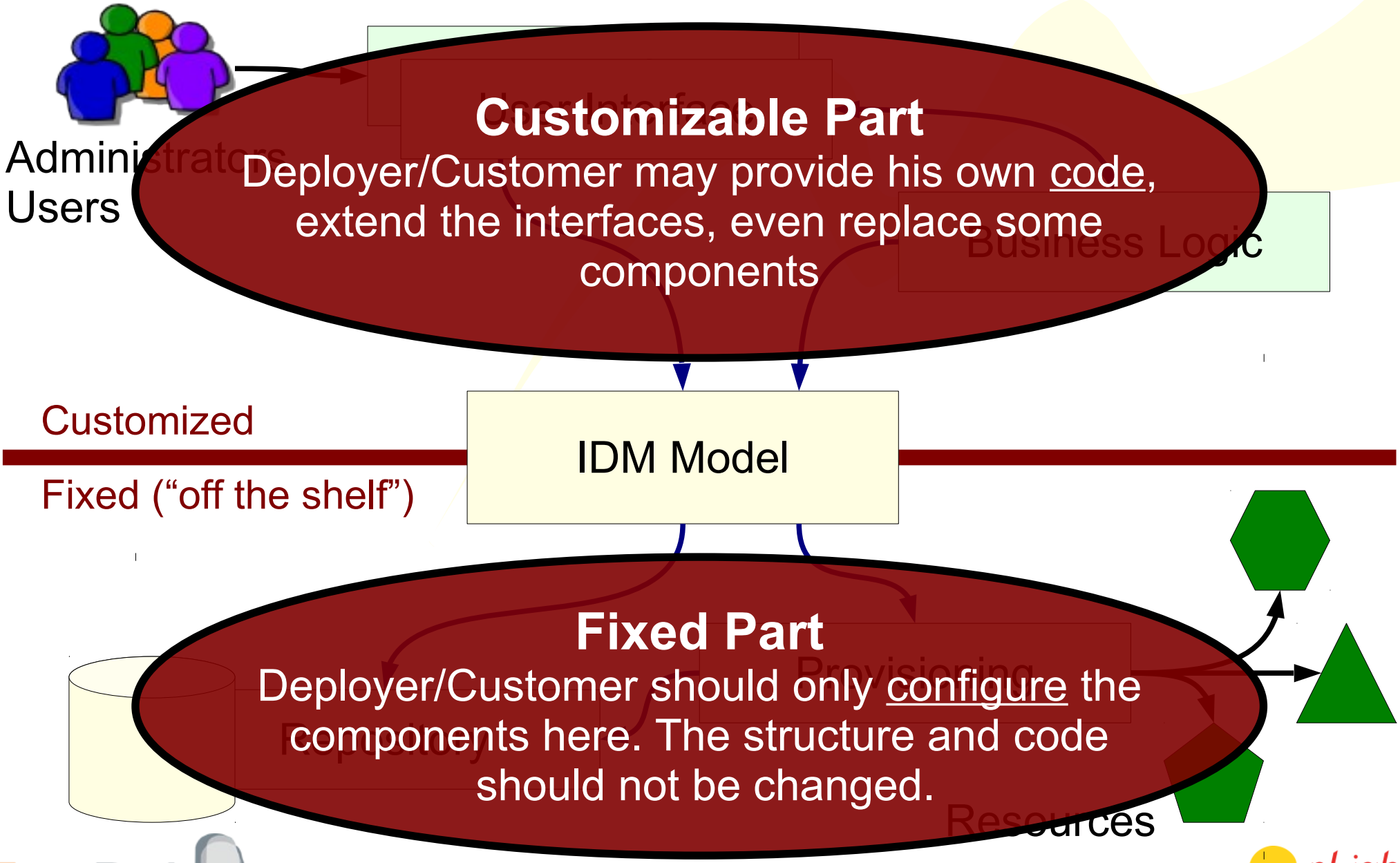
ForgeRock

nLight

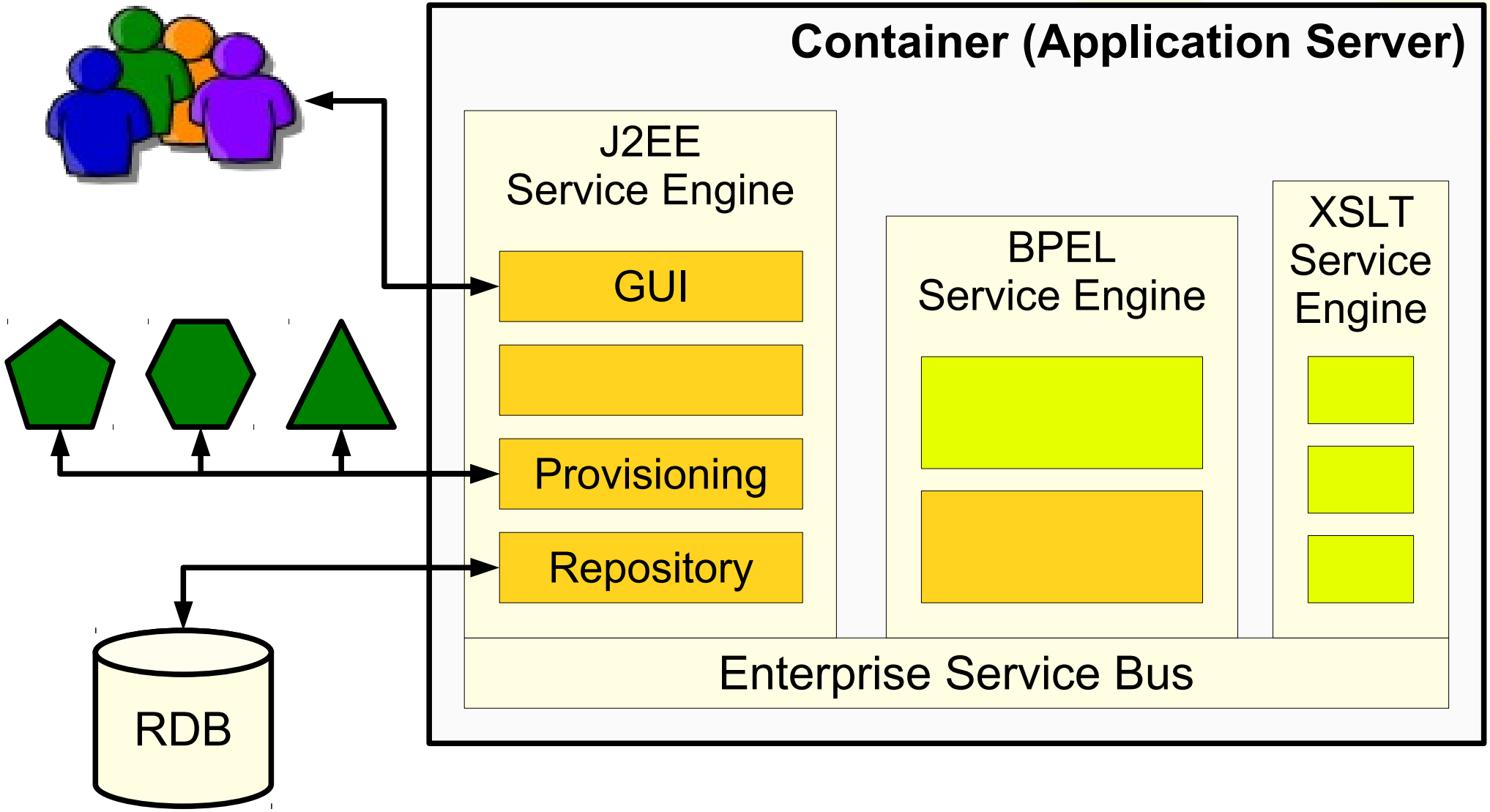
Abstract Architecture



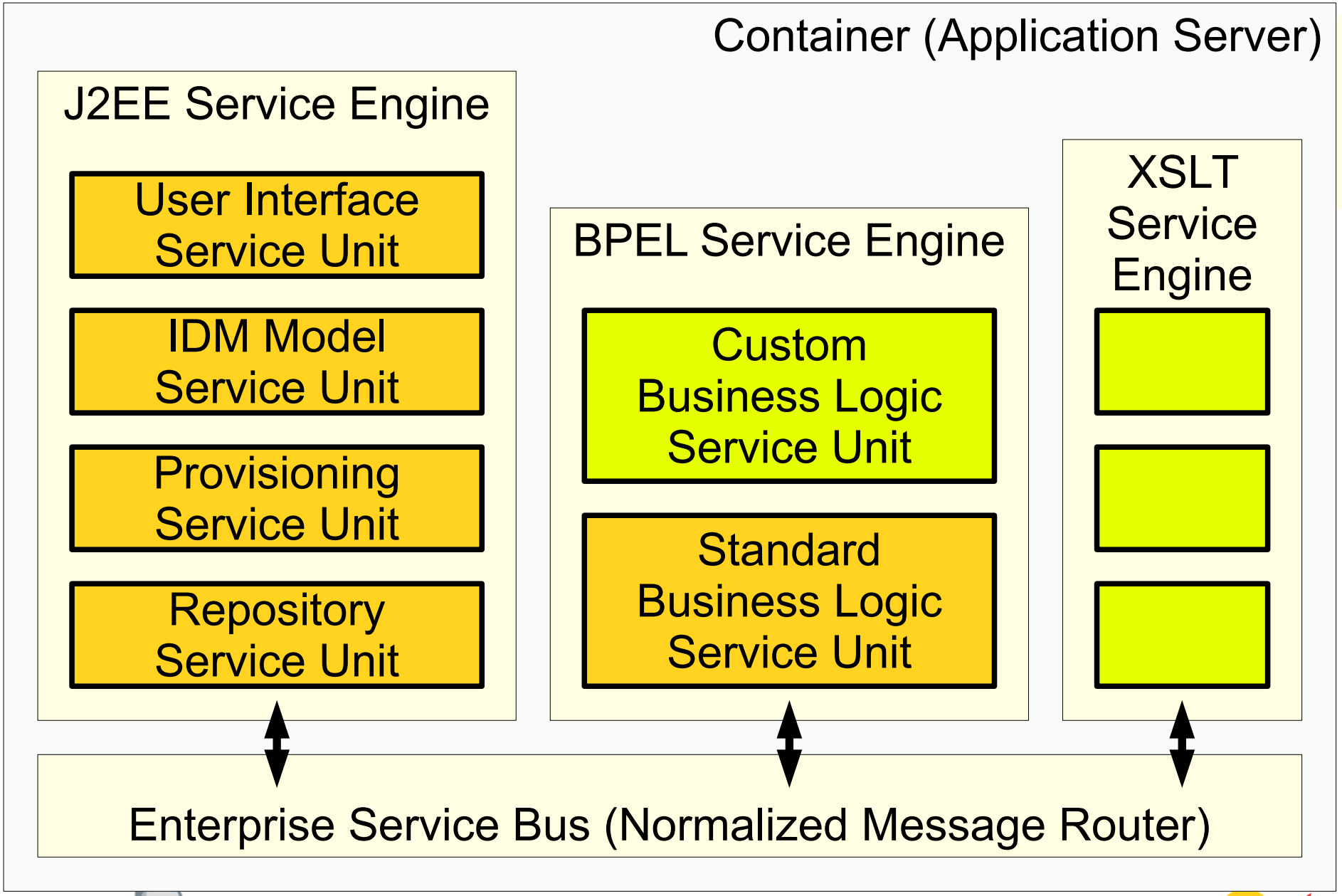
Abstract Architecture



Concrete Architecture (example)

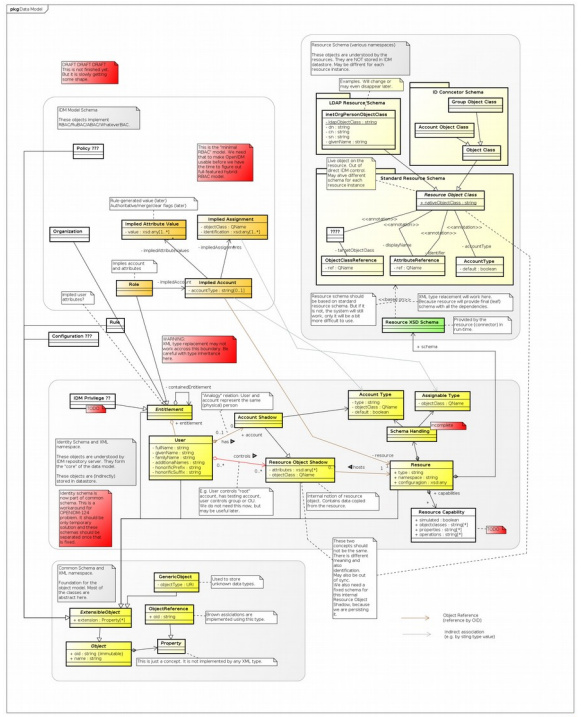


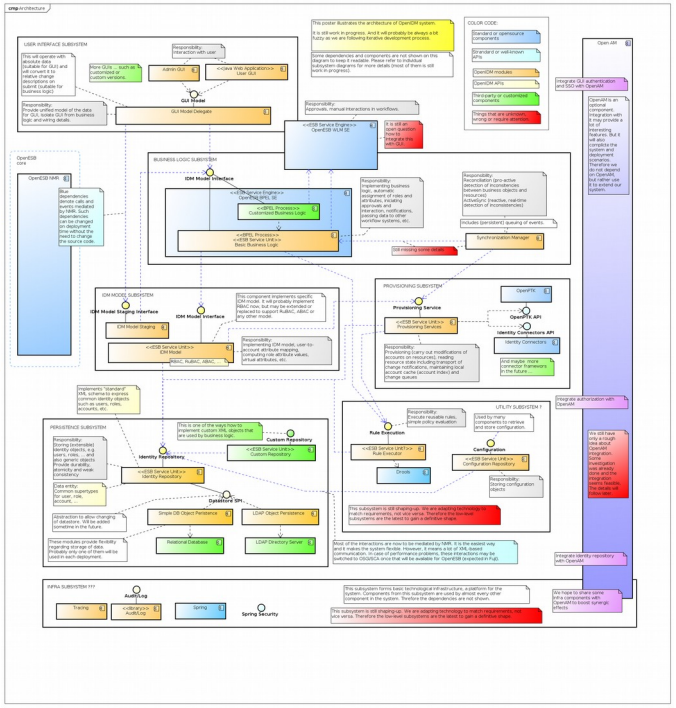
Concrete Architecture (example)

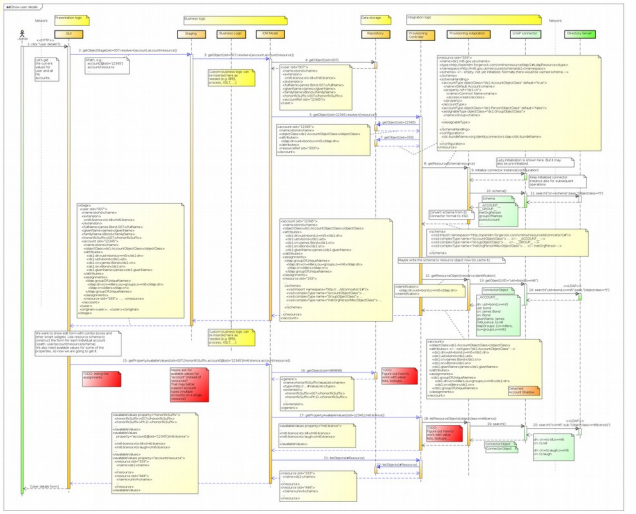


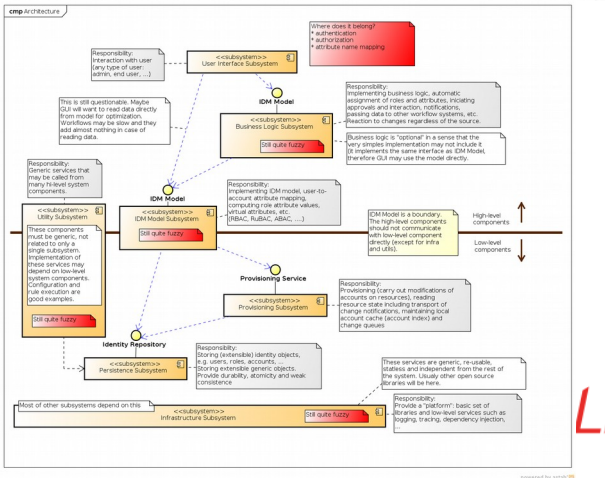
Even Bigger Pictures

- **UML Model (astah*)**
  - https://svn.forgerock.org/openidm/design

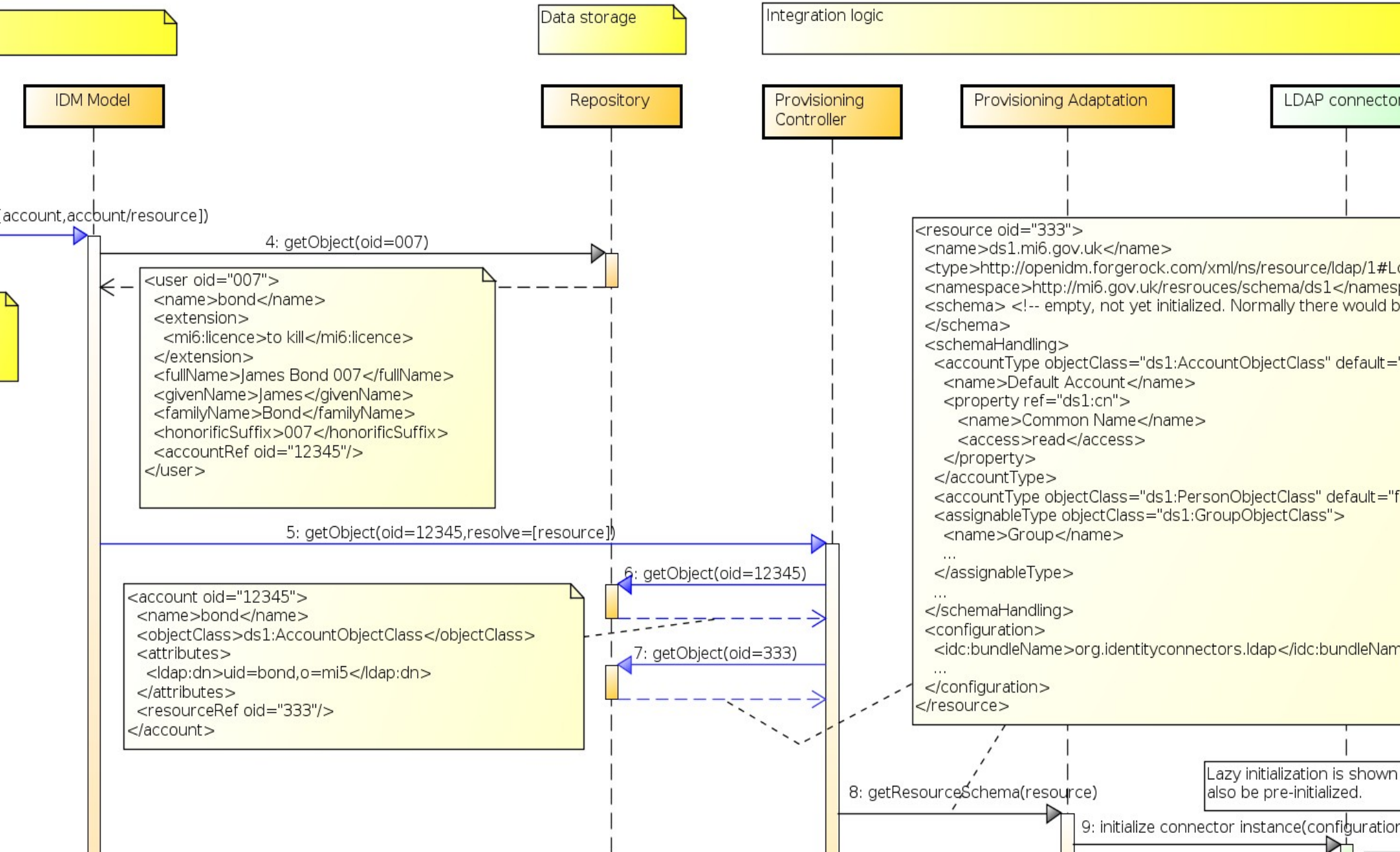








Even Bigger Pictures



(Much) More Information

- **OpenIDM Wiki**
  - https://wikis.forgerock.org/confluence/display/openidm/Home
- **[OpenIDM Architecture] in Wiki**
  - Wiki pages under [OpenIDM Architecture] page
  - “Live” architecture documentation
  - Includes UML diagrams
  - We try to keep it (reasonably) up to date
- **OpenIDM Mailing List**

OpenIDM Today

or What We Have Right Now

Current State

- **Architecture and almost all components in place**
- **Basic provisioning functionality works** (v1.6)
  - Platform: Glassfish/OpenESB v2 (for now)
- **Minimal automation**
  - Only synchronous BPEL workflows
  - Synchronization functionality almost done – testing (v1.7)
- **Advanced concepts roughly designed**
  - RBAC, Reconciliation, Auditing, Reporting

What Works

- **Read, create, update, delete of user and accounts**
- **Integration of Identity Connectors**
  - LDAP quite well tested, testing others right now
- **Synchronous BPEL processes**
- **Synchronization**
  - Testing it right now

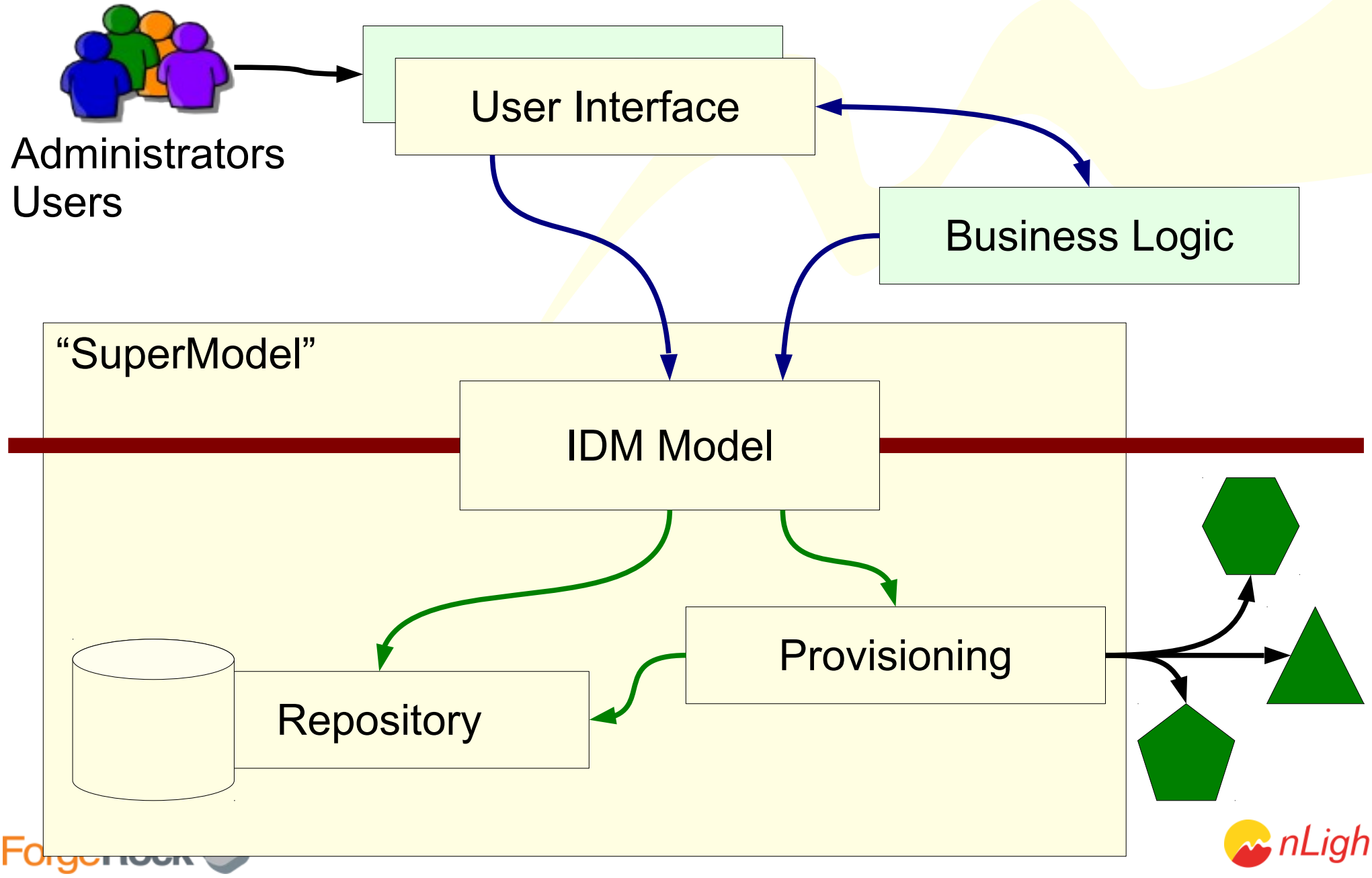
Releases

- **Snapshot 1.5: October 2010**
  - In fact a pre-release. Some basic provisioning
- **Snapshot 1.6: November 2010**
  - Basic “manual” provisioning (CRUD)
- **Snapshot 1.7: Real Soon Now ™**
  - Synchronization, Password Management

Some “Compromises” ...

- **OpenESBv2 and JAXB problems**
  - Slowing us down
  - Needed to change the deployment a bit

SuperModel Packaging



Some “Compromises” ...

- **OpenESBv2 and JAXB problems**
  - Slowing us down
  - Needed to change the deployment a bit
  - SuperModel packaging
    - May not be entirely wrong
    - In fact, this was quite expected

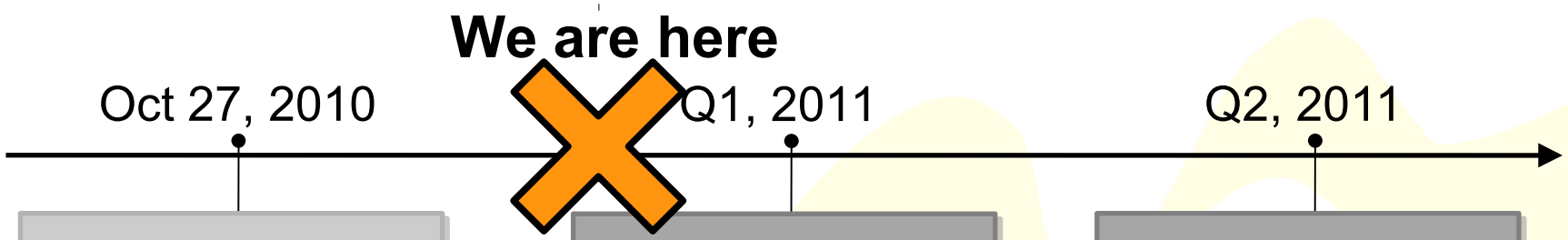
Some “Compromises” ...

- **Repository code is unmaintainable**
- **Some hardcoded things here and there**
  - Trying to keep track of them in Jira

Trunk: Always in a flux ...

- **There are always some bugs, bug fixing them quickly (hours)**
- **It is being stabilized right now**
  - The system is in testing now (feature freeze!)
- **Real usability expected soon (early 2011)**
  - Snapshot 1.7, Snapshot 2.0

Roadmap (2010-2011)



Snapshot 1.5

Features

- Core Architecture
- Integration with BPEL Workflow Service Engine
- Basic Provisioning and Account Administration

Core Framework for provisioning

Snapshot 2.0

Features

- Password Management
- Identity Synchronization with configurable Data Transformation Filters
- RACF Connector
- Tivoli Access Manager Connector
- End User Self-Service

ActiveSync and Password Management

Release 2.1

Features

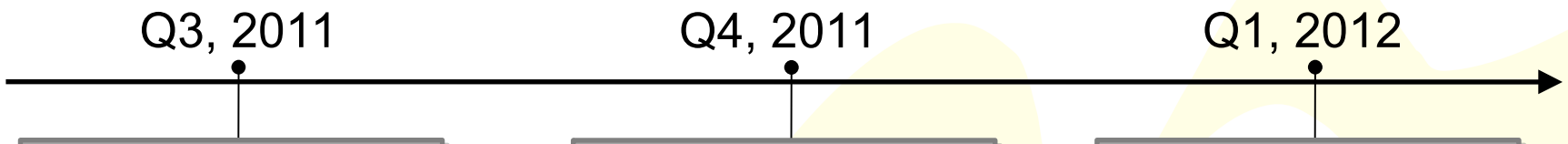
- Delegated administration
- Reconciliation
- Auditing & Reporting
- Provisioning Event Model with BPEL based workflows
- Role Based Provisioning
- User Interface Improvements
- OpenPortal Connector
- OpenAM Connector
- OSGi Based Deployment

Full featured Identity Management system

ForgeRock

nLight

Roadmap (2011-2012)



Snapshot 2.2

Features

- Compliant Provisioning and SoD policy engine
- Attestation on entitlement carrying attributes
- Integrated Role Engineering
- JMX Monitoring
- Event listener
- WS-SecureConversation support
- Additional Connector support

No need for separate Role Management solution

Snapshot 3.0

Features

- JSR-268 Portlets for Self-Service and Account Administration
- Asynchronous provisioning
- Event tracking and statistic reporting
- Fine grained authorization (OpenAM Snapshot 12 (Q3 2011) Entitlements enhancements)
- Upgrade tool

Decoupled UI for easy Portal Deployment

Snapshot 3.1

Features

- Appliance deployment mode
- Migration and upgrade tool enhancements
- Open Source business intelligence integration
- Full SOA integration capability with external modules/vendors (OpenAM Snapshot 13 (targeted Q4 2011) Session impersonation and shadowing)
- Architecture

Identity & Access Intelligence

Forge

nLight

OpenIDM Tomorrow

or The Lessons Learned

Immediate TODO

- **Repository needs rewrite**
  - Current repository: Result of a wild mix of misunderstanding and lack of time
- **Provisioning needs cleanup**
- **Maven components need review**
  - cleanup package and project names
- **Source tree structure documentation!**

Design TODO

- **Packaging & Deployment**
  - ESB, OSGi, OpenESB, …
- **Authentication/Authorization**
  - Admin GUI and End User GUI
- **RBAC**
- **Asynchronous things**
  - Approvals, async provisioning, …
- **Auditing & Reporting**

Smoother Development

- **Thou shalt not break the build!**
  - svn up; mvn clean install; svn commit
- **Document motivation** (in the code)
  - Why it was done like that?
- **Component maintainers**
  - Nobody can know everything, split the load
- **Using Jira**

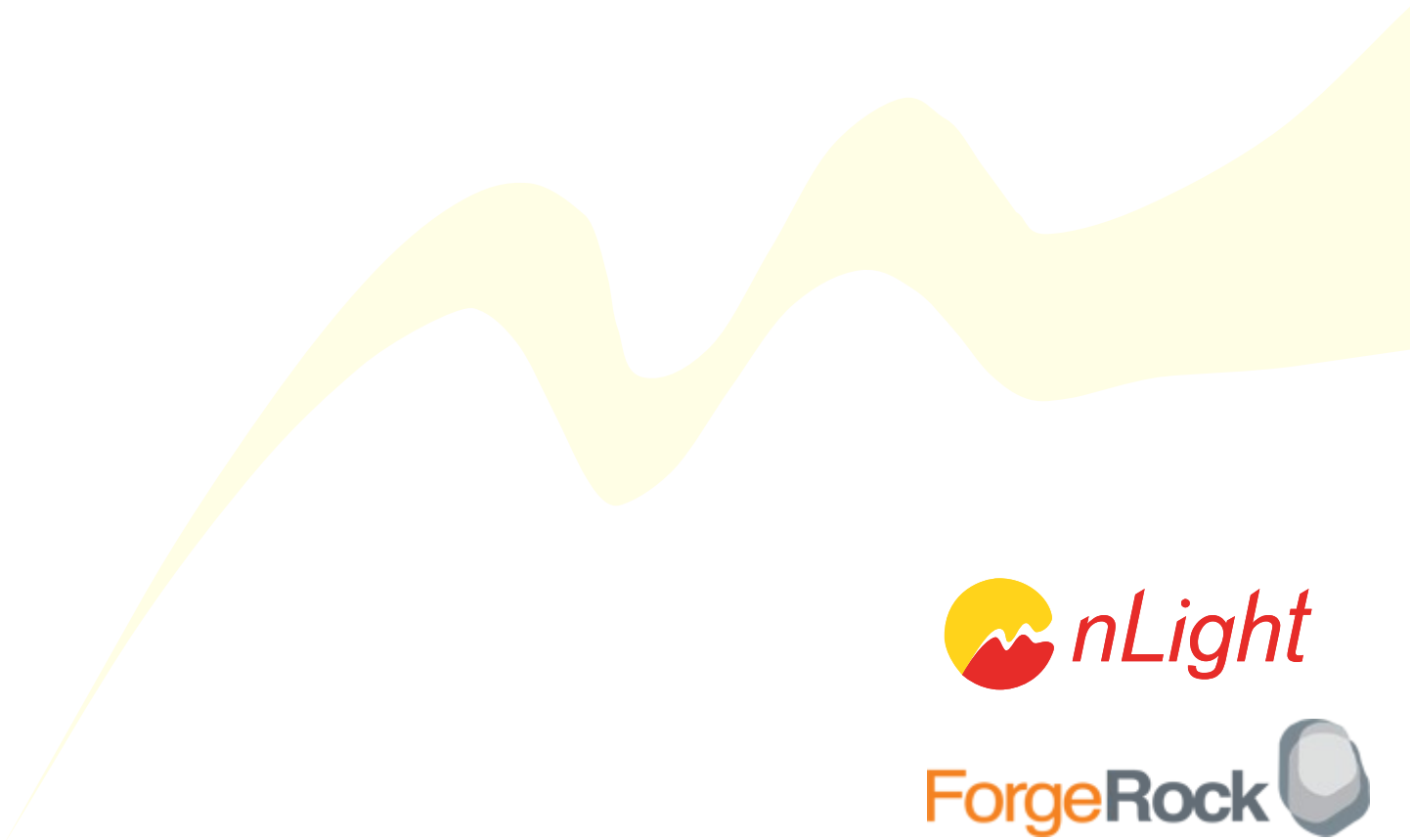
Need to ...

- **Open up the communication**
  - IRC? Mailing List?
- **Improve documentation**
  - I don't want to be the only writer here
- **Open up the design process**
  - How? Better participation from others
  - Reviews? Comments? … preferably before it is too late

Need ...

- **Release early, release <u>often</u>**
  - Regular and predictable release cycle
  - Be more SCRUM-like?
- **More (automated) testing**

Conclusion



OpenIDM Yesterday

- **Yesterday**
  - Good idea, good design, some code, something works
- **Today**
  - Building up, the system grows, starts to be usable
  - … yet there are some problems
- **Tomorrow**
  - Time to improve the code and regain speed
  - Time to improve the efficiency of development process

Overall ...

- A lot of work has been done
- It (mostly) works!
- We are going in the right direction
- Just need to do small adjustments
- … let's rock!

Questions and Answers

Thank You

Radovan Semančík

N-Light, s.r.o.
Vendelínska 109
900 55 Lozorno
Slovakia

EXTRA SLIDES

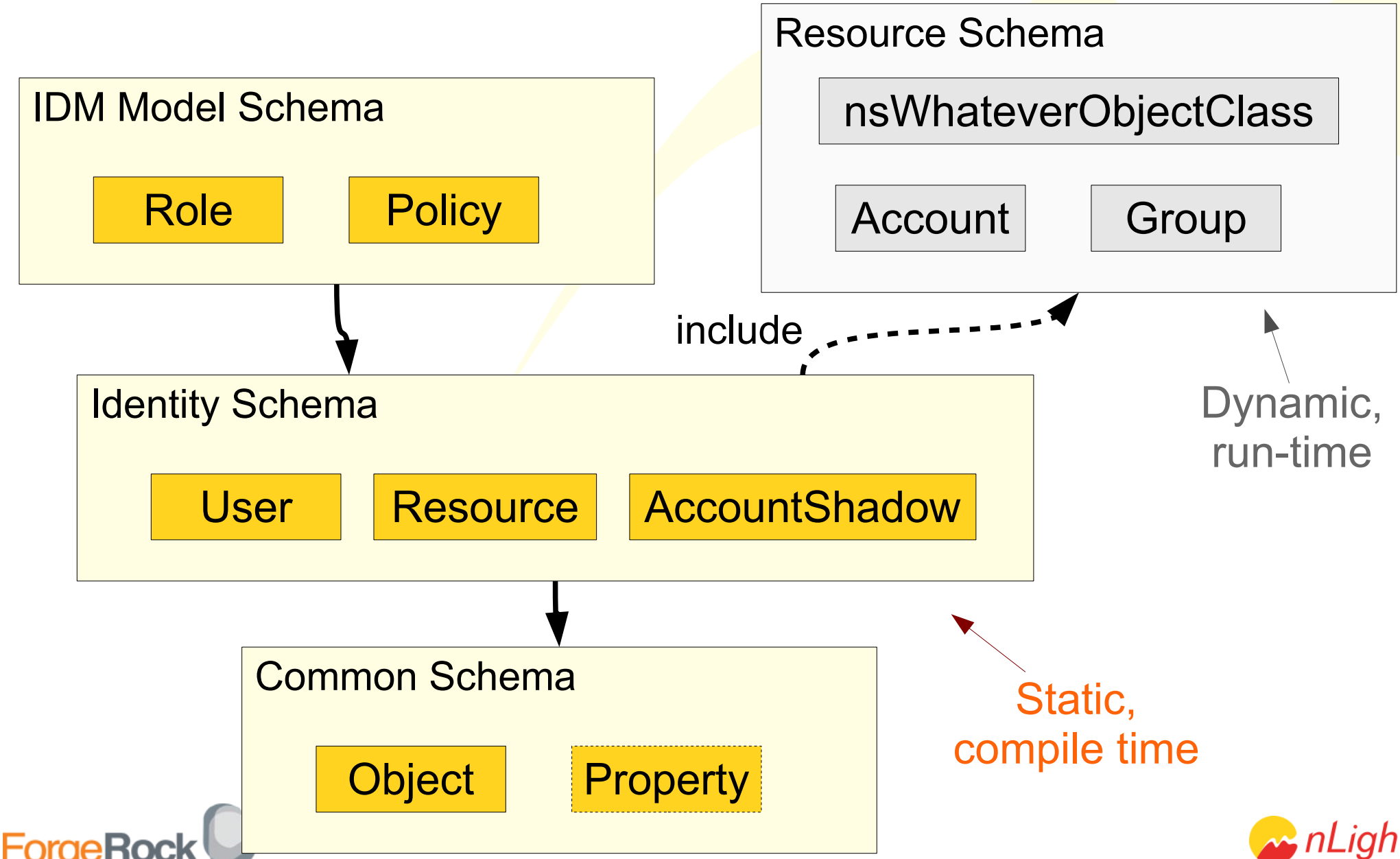
Data Model

nLight

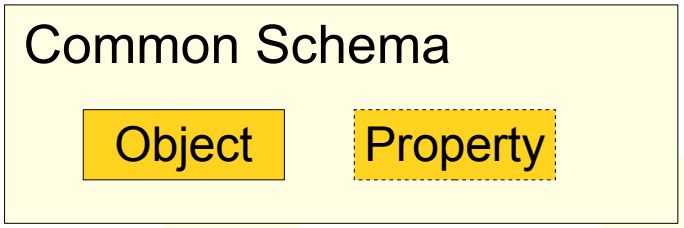
Data Model

- **Based on XSD schemas**
- **Extensible**
  - Using both XML type replacement and xsd:any
- **Separated to several partitions**
  - Common Schema
  - Identity Schema
  - IDM Model Schema
  - Resource Schema

Data Model Overview



Common Schema



- **Common concepts**
  - Object, Extensible Object, Generic Object, Property
  - Object changes, property references, …
- **Simple and generic**
  - Not specific to IDM problem domain
  - All components can understand the concepts from this schema

Objects and OID

- **Object (`ObjectType`)**
  - Everything that we can store and process is an Object
- **Objects are uniquely identified by OID**
  - OID = Object Identifier
  - Persistent, immutable, non-reassignable
  - Most likely UUID with optional domain suffix

Properties

- **Object have properties**
  - Properties are usually first-level XML elements

```
<user oid="007-2345-394728234-398540565">
  <name>bond</name>
  <fullName>James Bond</fullName>
  <givenName>James</givenName>
  <familyName>Bond</familyName>
  …
</user>
```

Properties

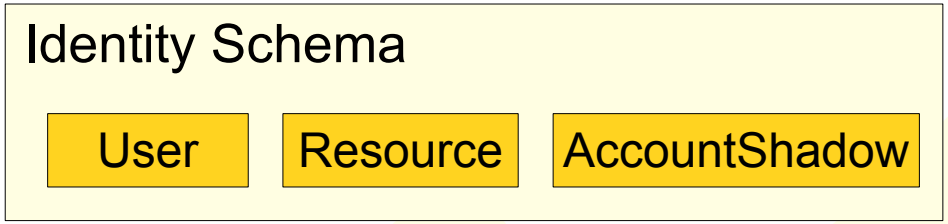
- **Properties are atomic**
  - They can be added, removed replaced – as a whole
  - Cannot be modified “inside”, only replaced
  - Properties are the “finest grain” that the system cares about (atoms)
  - The system does not understand (and does not care) about internal structure of a property

Objects and Properties

- Properties can be arbitrarily complex

```
<imaginaryObject oid="123456">
  <name>foobar</name>
  <fullName>Foo Bar</fullName>
  <foo:geekName>F00 B4r</foo:geekName>
  <pet:pet species="dog" breed="basset"
       name="Doggie"/>
  <com:shoppingPreferences>
      <com:tShirt size="XXL" color="#000000"/>
      <com:tie preference="no thanks"/>
  </com:shoppingPreferences>
</imaginaryObject>
```

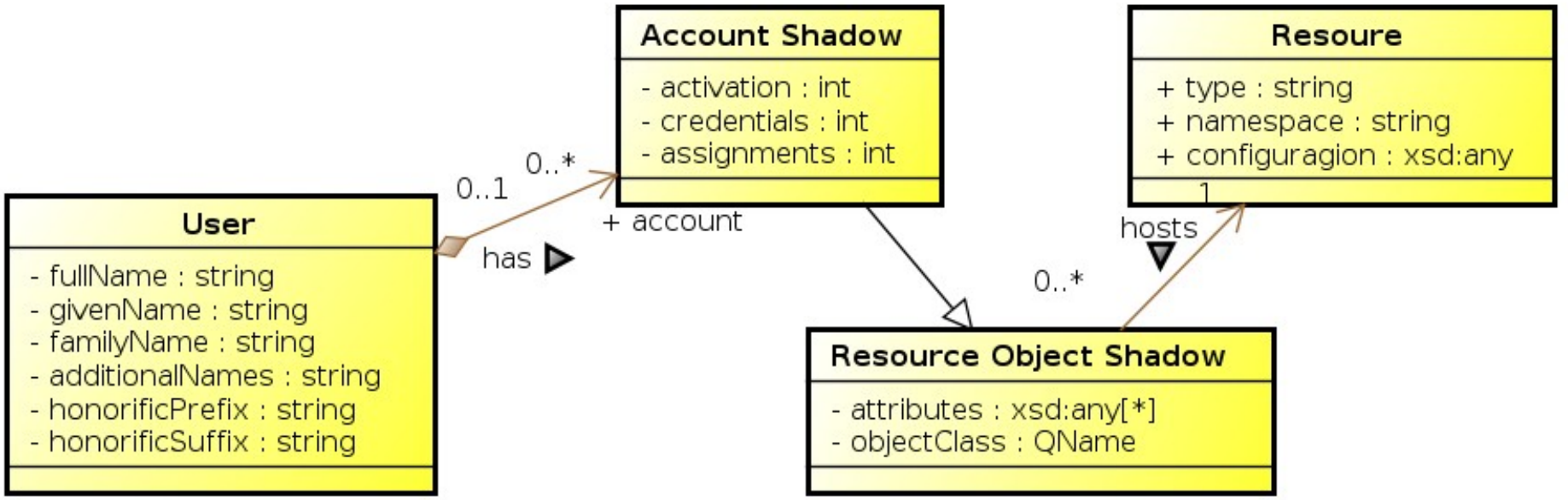
Identity Schema



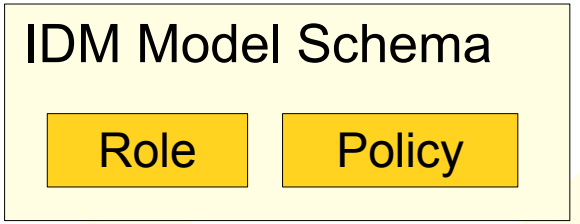
- **IDM-specific concepts**
  - User, Resource, ResourceObjectShadow, AccountShadow, Entitlement, …
- **Domain data model (“core”)**
  - Understood by provisioning (partially)
  - Fully understood by IDM Model and all components above
- **Reusable**
  - Independent of access control model and business logic

Basic Identity Schema Concepts

(simplified)



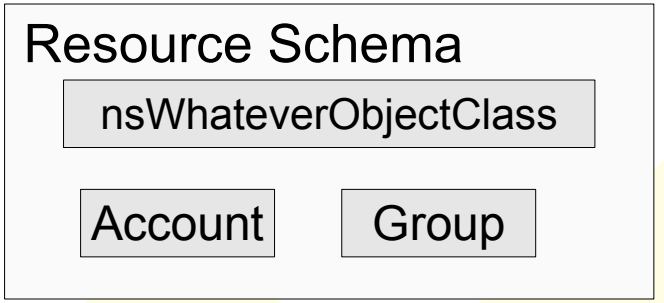
IDM Model Schema



- **Advanced IDM concepts**
  - Roles, Rules, Policies, Security Controls, Labels, ...
  - … or any other concept that is needed ...
- **Business-oriented**
  - Business logic (and GUI) is using these concepts
- **Theoretically interchangeable**
  - But the replacement won't be easy

WORK IN PROGRESS

Resource Schema



- **Resource-specific concepts**
  - Account, Group, nsRole, groupOfUniqeNames, location, department, … almost anything …
- **Specific to resource instance**
  - E.g. two Sun DSEE 6.3.1 servers may have different LDAP schema, therefore different resource schema
- **Available at run-time only**

Resource Schema Example
(idealistic)

```
<object xsi:type="ldap:InetOrgPersonObjectClass">
    <ldap:dn>uid=bond,o=mi5</ldap:dn>
    <ldap:uid>bond</ldap:uid>
    <ldap:cn>James Bond</ldap:cn>
    <ldap:sn>Bond</ldap:sn>
    <ldap:givenName>James</ldap:givenName>
    <ldap:ou>Agents</ldap:ou>
    <ldap:ou>Shooters</ldap:ou>
</object>
```

Resource Schema Example
(realistic)

```
<object xsi:type="ds1:AccountObjectClass">
   <ids:uid>12386-238947-349-209348</ids:uid>
   <ids:name>uid=bond,o=mi5</ids:name>
   <ds1:uid>bond</ds1:uid>
   <ds1:cn>James  Bond</ds1:cn>
   <ds1:sn>Bond</ds1:sn>
   <ds1:givenName>James</ds1:givenName>
</object>
```

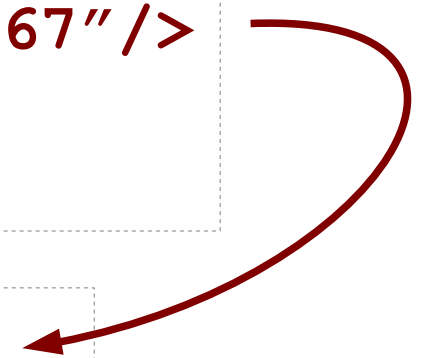
Implementation details of Identity Connectors Framework leak into the schema (ids: namespace above)

Object References

- **Implement associations between objects**
  - E.g. account belongs to a user

```
<user oid="007-7777-777”>
  <name>bond</name>
  ...
  <accountRef oid=”1234-4567”/>
  ...
</user>
```

```
<account oid="1234-4567”>
  <name>jbond</name>
  ...
</account>
```



Composite Objects

- Alternative way how to represent associations

```
<user oid="007-7777-777”>
  <name>bond</name>
  ...
  <account oid="1234-4567”>
    <name>jbond</name>
    ...
  </account>
  ...
</user>
```

- Similar to “view” in Sun IDM, but cleaner, unified principle ... and more flexible.

Object References and Composite Objects

- **Easy to convert one to the other**
- **Object references are used in the repository**
  - Normal database forms, store object in one place
- **Composite objects used in business logic**
  - But references are also occasionally used in business logic (for optimizations)

Pure XML

- **Everything is pure XML and XSD**
  - No meta-schema or schema-on-top-of-schema
  - Property types defined by schema
    - xsd:simpleType, xsd:complexType
  - Single/mutli-valued properties defined by schema
    - minOccurs, maxOccurs
  - Using XSD annotations to extend XSD when needed
- **We try to avoid hacks as much as we can**

Extensibility

- **Extending existing object type**
  - Using <extension> element
- **Creating new object type**
  - Using `GenericObjectType`
    - Practical, but slightly inconvenient
  - Using XML type replacement
    - Theoretical, problematic

Extending Existing Objects

- **<extension> element**
  - All “normal” objects have <extension> element of xsd:any type
    - Defined in `ExtensibleObjectType` supertype
  - <extension> may contain any non-standard properties
- **Ignored by low-level system components**
- **May be used by business logic**

<extension> Example

```
<user oid="d3ad-m3a4-4758-39488740" version="42">
  <name>foobar</name>
  <extension>
    <foo:geekName>F00 B4r</foo:geekName>
    <org:guild>Societus Geekus Europeus</org:guild>
    <org:guild>Basset User Group</org:guild>
  </extension>
  <fullName>Foo Bar</fullName>
  <givenName>Foo</givenName>
  <familyName>Bar</familyName>
  ...
</user>
```

Subsystems

nLight

User Interface Subsystem

- **End User and Administration interfaces**
- **Based on JSF2/IceFaces**
  - "Web 2.0", RIA, AJAX, … and similar buzzwords
- **Should provide reasonable functionality**
  - Smart components to display well-formatted data based on XSD schema
- **We expect customization and extension**

User Interface Customization

- **Now: We are open source**
  - Just get the sources and tweak them
- **Later: Form objects or artifacts to customize the UI without programming**
- **UI can be replaced or substantially extended**
  - E.g. replace End User interface with portlets
  - UI is built on top of well-defined interfaces

Business Logic Subsystem

- **“Empty” by default**
- **ESB-based extensibility**
  - Well-defined interface of IDM Model
- **Lot of options to choose from (service engines)**
  - BPEL – the usual suspect
  - Java (J2EE, POJO) – the classic
  - XSLT – for simple things
  - …

IDM Model Subsystem

- **Provides unified façade to User Interface and Business Logic**
  - It is a boundary between high-level and low-level part of the system
- **Processes policies, Role-based models (RBAC), virtual attributes, …**
  - … whatever that can be reasonably formalized to a model

IDM Model Subsystem

- **Managed Object Types**
  - Role
  - Rules (most likely, but that is still TBD)
  - Policy
  - …

Note: This is still work in progress.

Default IDM Model

- **The IDM Model that will be shipped with OpenIDM**
  - Will influence the User Interface and most deployments
  - Should support 80% of cases with 20% of complexity
- **Still TBD, but most probably hybrid RBAC model**
  - Roles (hierarchical) combined with rules

Null IDM Model

- **Does "nothing"**
  - No roles, no policies, no virtual attributes … nothing
- **Expects that business logic does everything**
- **Can be used in heavily-customized deployments**
  - … and also for testing!

IDM Model Replacement

- **IDM Model can theoretically be replaced**
- **But such replacement will be difficult**
  - User Interface depends on the model
  - Business Logic depends on the model
- **Still may be needed for some extreme deployments**

Provisioning Subsystem

- **Integration Logic**
  - Business logic should **not** be here
- **Communicates with other systems (resources)**
  - Protocol adapter, schema converter
- **Describe the options, but do not decide**
  - List object classes, describe schema, ...
  - It is the IDM Model and Business Logic that decides

Provisioning Subsystem

- **Managed Object Types**
  - Resource Object Shadow (and subtypes)
  - Resource
- **Pluggable and configurable design**
  - We need great deal of flexibility there
  - But we want to avoid programming
    - This is targeted at operations staff (admins), not developers

Identity Connectors Framework

- **Sun open-source framework**
- **The architecture is not ideal**
  - We will need an adaptation layer on top of it
- **The future under Oracle is questionable**
- **Supports a good set of resources**
- **We will use it, at least for now**
  - But we will need a long-term solution

Persistence Subsystem

- **Provides object repository**
  - Storing, retrieving and modification of objects
  - Atomicity, consistency – some details still TBD
- **Based on relational database now**
  - JPA/Hibernate
- **Can support even different stores later**
  - Especially LDAP
- **Possible integration with OpenAM**

Open and Dynamic Development

- **Completely open development**
  - Public task tracking (Jira)
  - Public communication (mailing lists, wiki)
  - Public planning (roadmap, Jira)
- **User (customer) participation**
  - Customer needs take precedence
  - The roadmap can be adapted to match real needs

Open Discussion

nLight

Complications and Open Questions

- **IDM Model design**
- **Workflow persistence**
- **Human interaction (approvals)**
  - … and long-running business processes
- **Asynchronous provisioning**
- **Replacement for Identity Connectors**

Extra Slide: Notes and TODO

- **TODO**
  - Describe synchronization and reconciliation
  - More about RBAC