Evolveum®

Fighting Software Maintainability Nightmares

Radovan Semančík

Lecture at Technical University of Košice

March 2019

Radovan Semančík

- Software Architect at Evolveum
- Architect of midPoint
- Apache Foundation Committer
- Contributor to several open source projects

Software Maintainability

- So, you have written your little software today

  … that’s nice

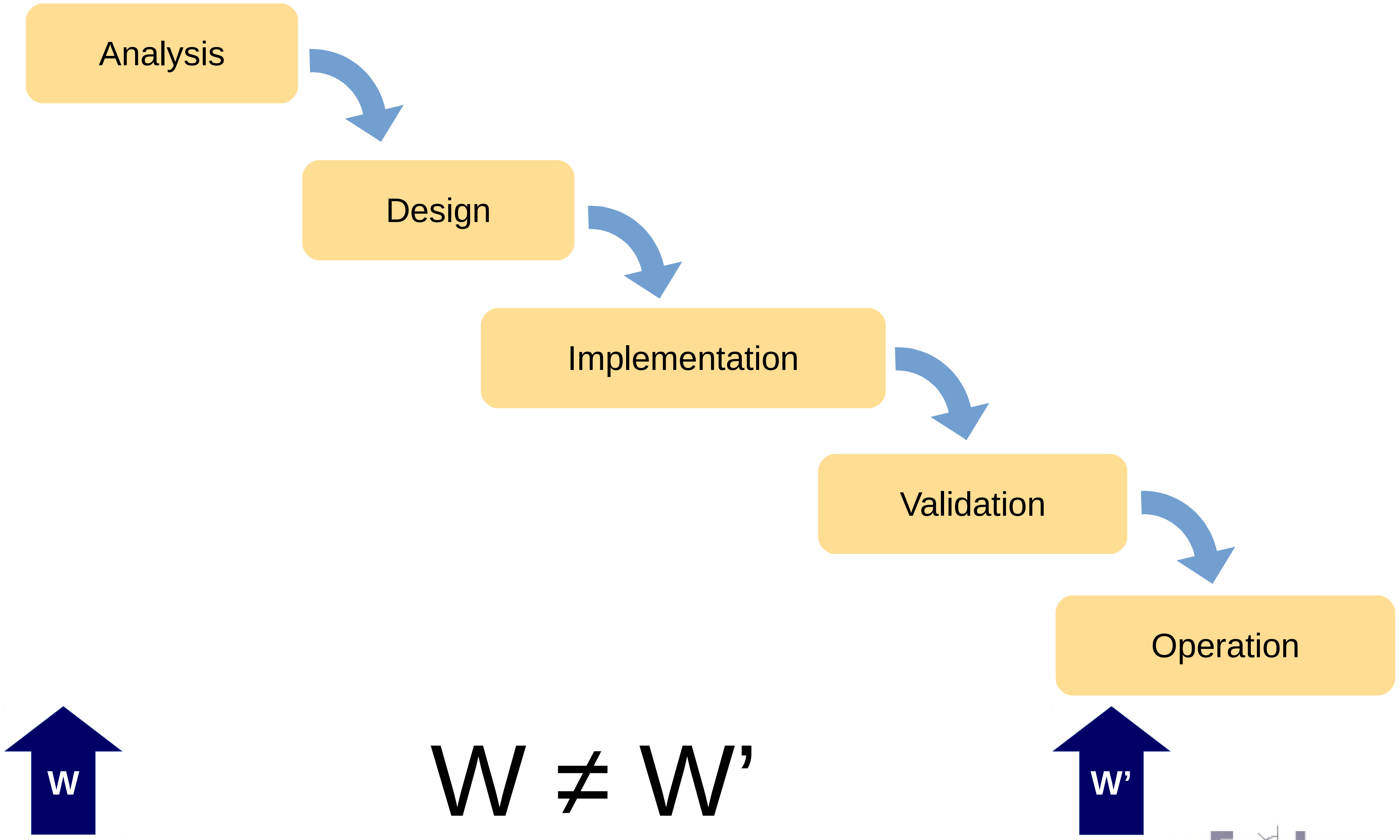
- Little software will grow and grow and grow
- Because **software is never done**
- Software must change, adapt, evolve
- Can you keep your software alive?
- For a year? Or 5 years? Or 10 years?

Software Maintainability **Nightmare**

- The day when your software is deployed is the **first** day of its life span, not the last one.

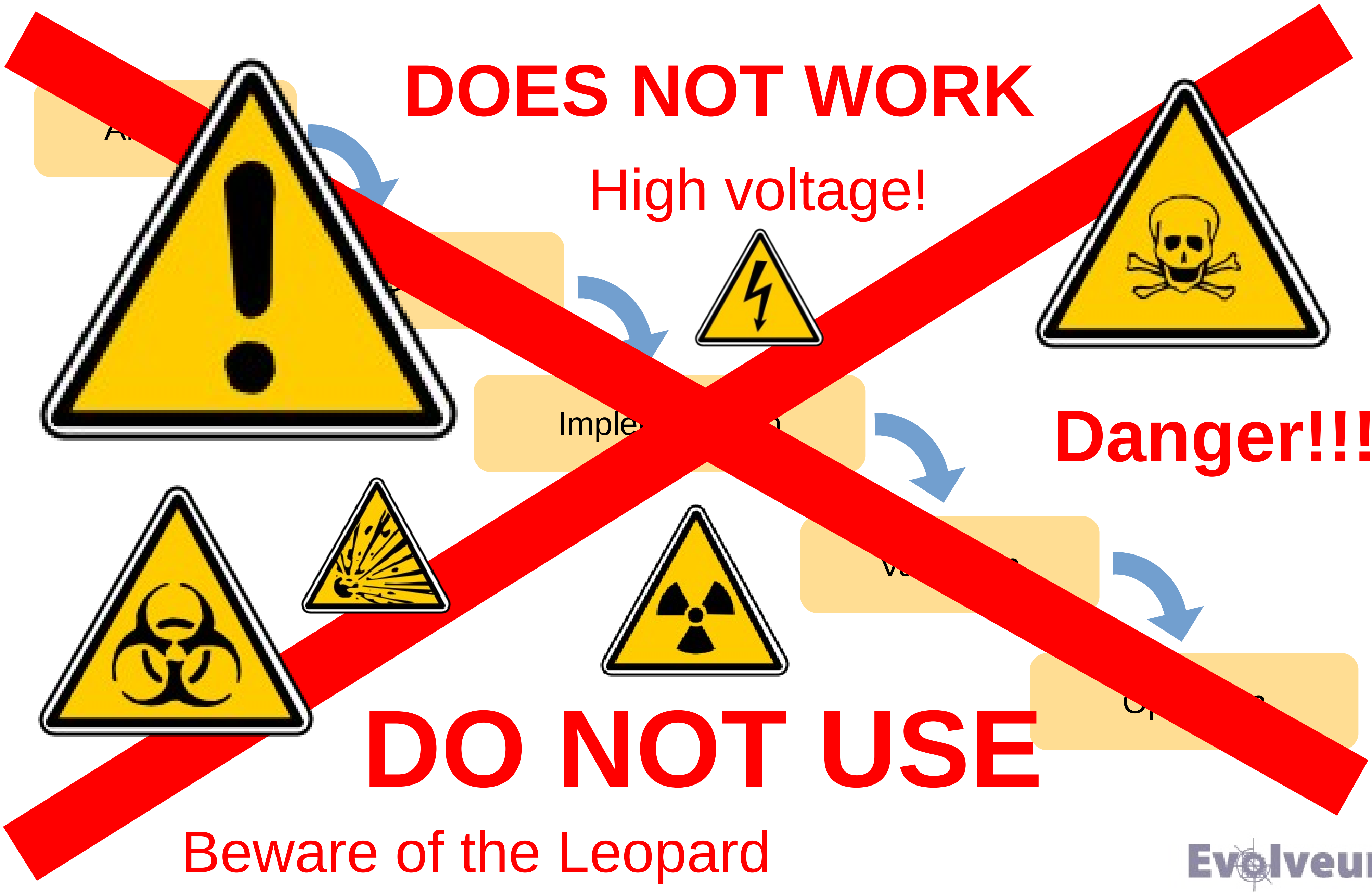
- It is **hard** to write the software. To make it run.
- It is **much harder** to keep it running.

Waterfall Model



W ≠ W'

Waterfall Model



DOES NOT WORK

High voltage!

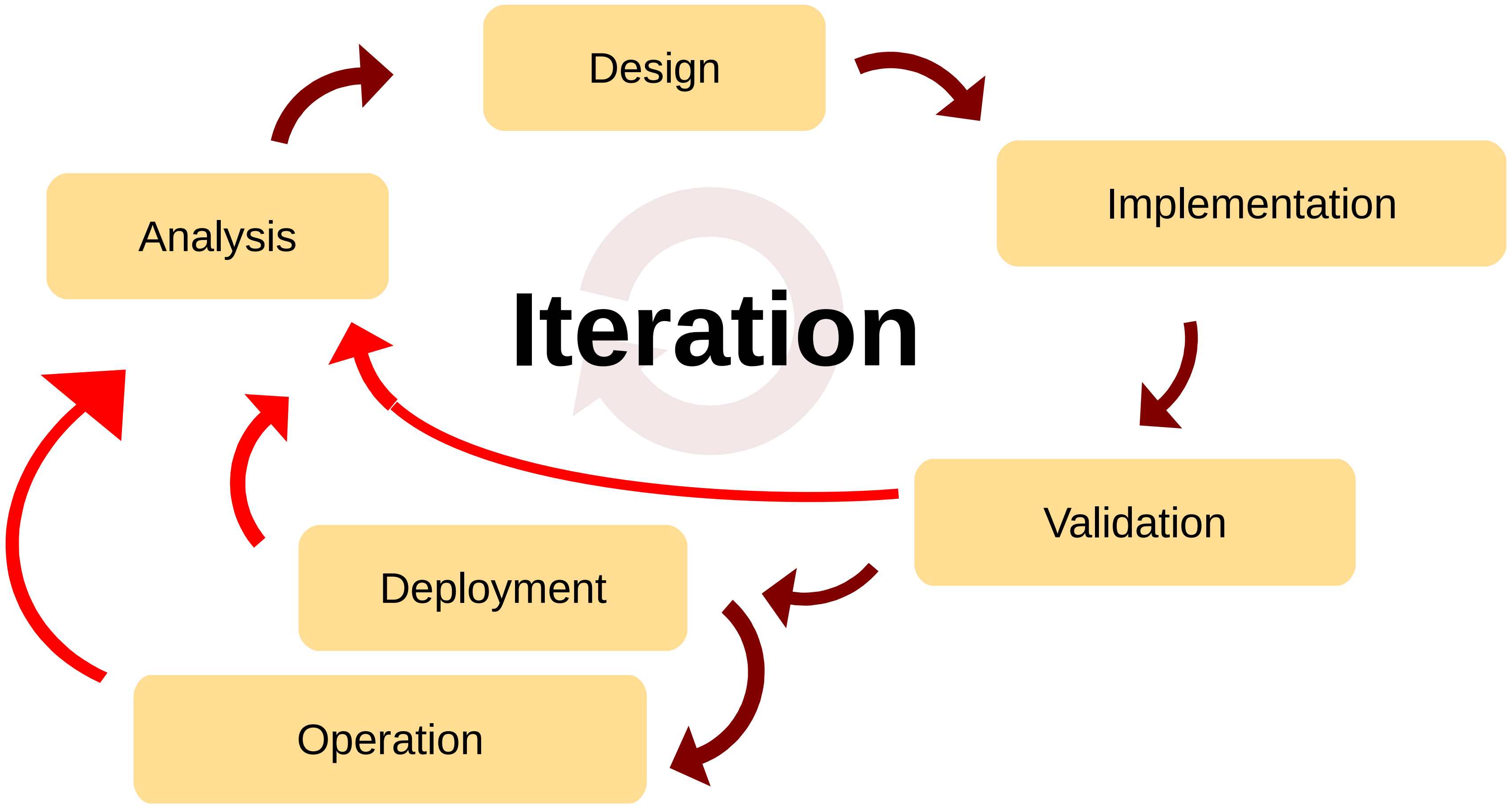
Danger!!!

DO NOT USE

Beware of the Leopard

Evolveum

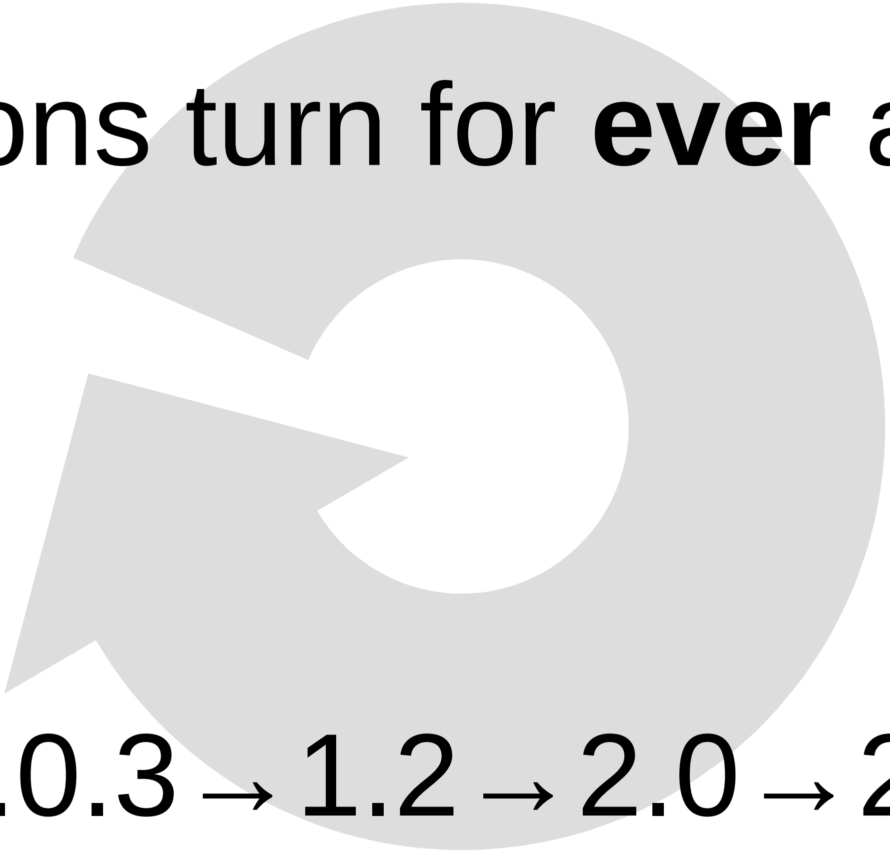
Iterative Models



- Works (much) better … because it works at all
- But **world will not stop changing** after deployment

Reality

Iterations turn for **ever** and **ever**



→ 0.1.2 → 1.0.3 → 1.2 → 2.0 → 2.0.1 → 2.1 → 2

Software Maintainability **Nightmare**

- Correctness
  - Do the same thing, do it right (bugfixes)
- Security
  - Do the same thing, but securely (security updates)
- Adaptation
  - Do the same thing, but in a changed world
- Continuity
  - Do the same thing, but in new version (upgrades, retention)
- Evolution
  - Do more and better things (new features)

Software Maintainability **Nightmare**

- Correctness
  - Do the same thing, do it right (bugfixes)
- Security
  - Do the same thing, but securely (security updates)
- Adaptation
  - Do the same thing, but in a changed world
- Continuity
  - Do the same thing, but in new version (upgrades, retention)
- Evolution
  - Do more and better things (new features)

… it takes all the running you can do, to keep in the same place.

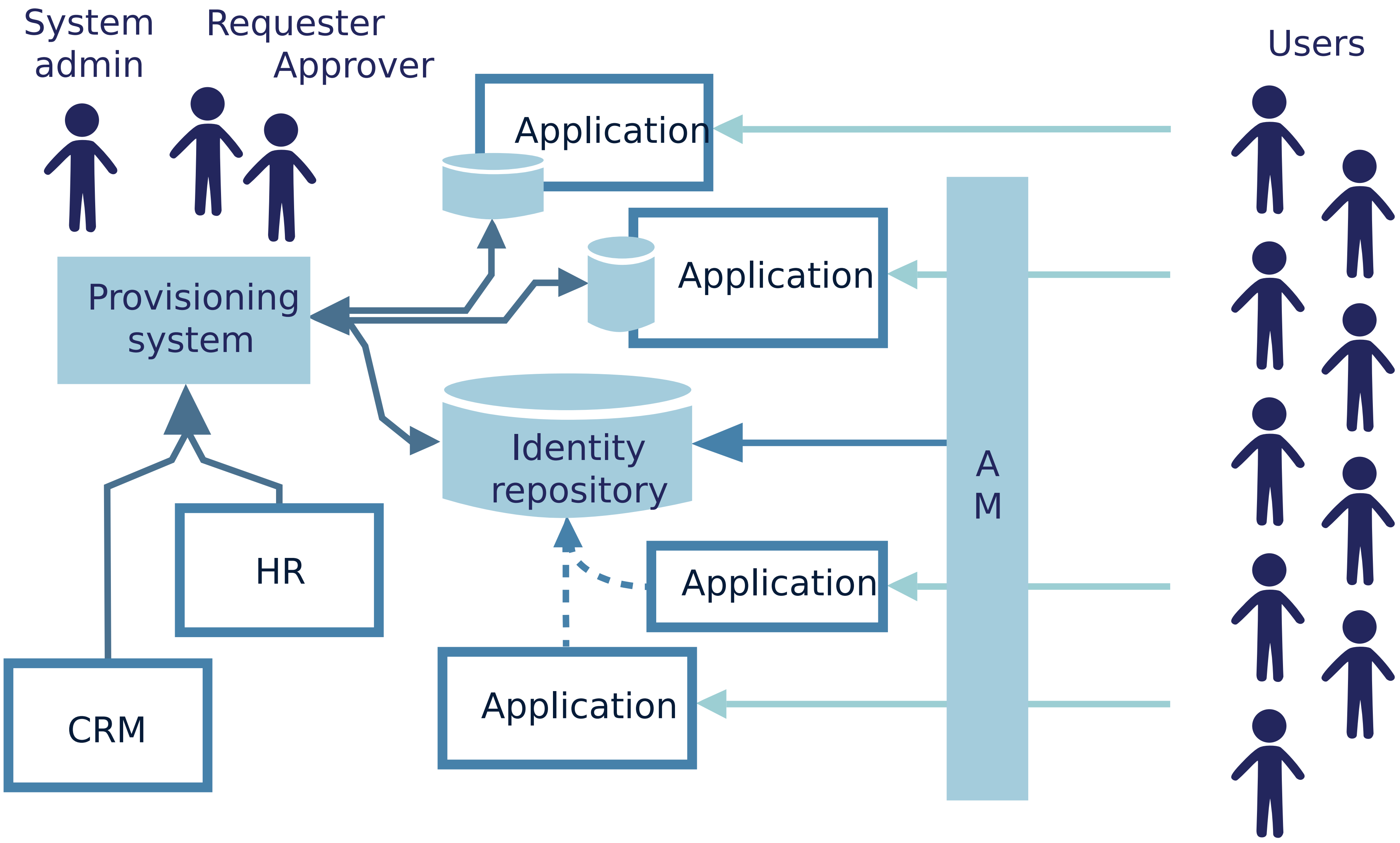
– Red Queen

Who are you anyway?
How dare you talk like this?

Project midPoint

- Identity management and governance
- Open source (Apache License)
- Started in 2011 by Evolveum (self-funded)
- Approx. 1 million lines of code
- Mostly written in Java

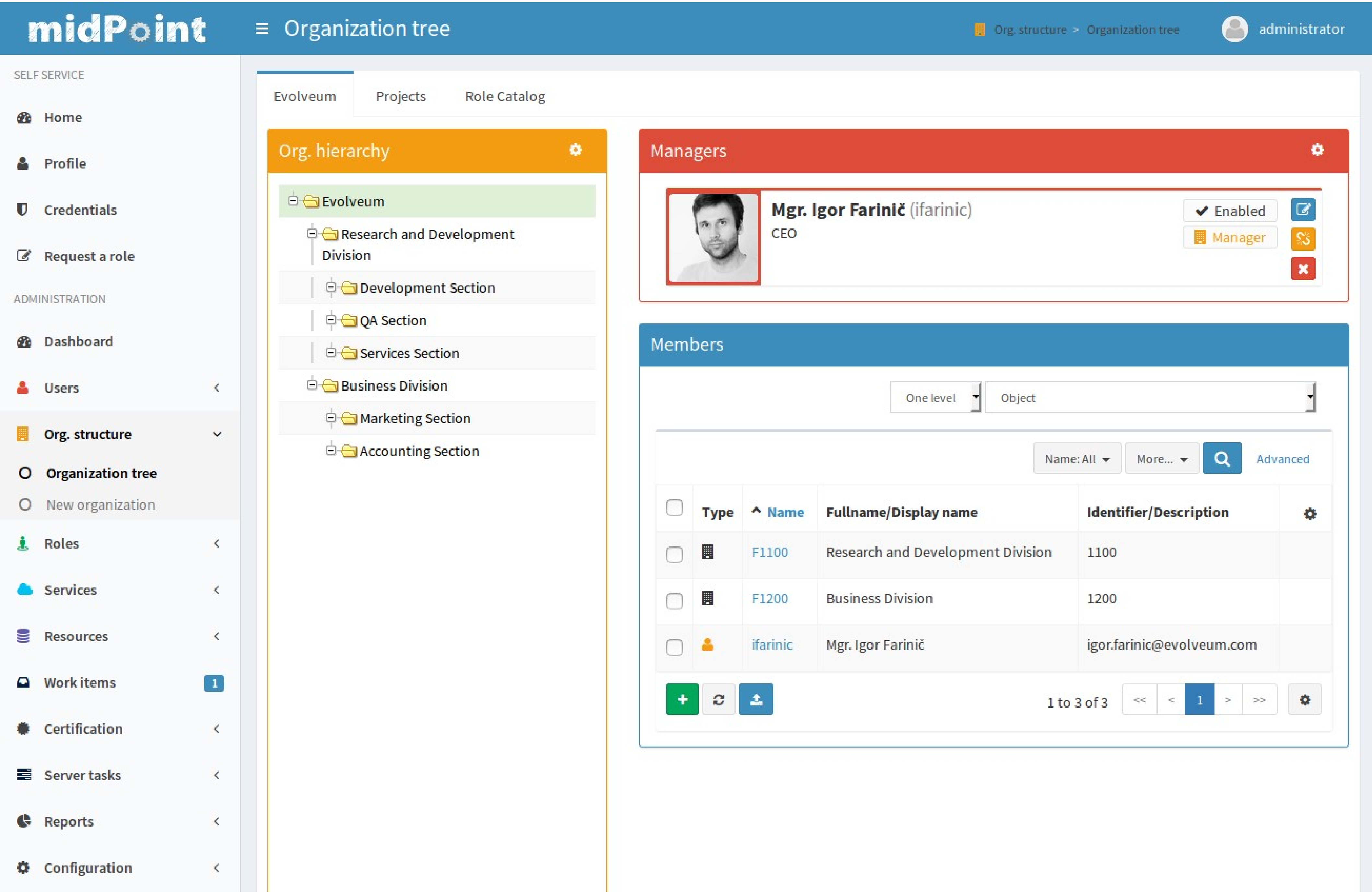
What is Identity Management?



Evolveum

… and Identity Governance?

- Beyond Role-Based Access Control (RBAC)
- Organizational structure
- Delegation, Audit, etc.
- Role assignment and re-certification
- Policies (e.g. SoD)
- Maintenance of role model (role lifecycle)
- Risk assessment
- Compliance


midPoint ≡ Organization tree

Org. structure > Organization tree administrator

SELF SERVICE

- Home
- Profile
- Credentials
- Request a role

ADMINISTRATION

- Dashboard
- Users
- Org. structure
  - Organization tree
  - New organization
- Roles
- Services
- Resources
- Work items 1
- Certification
- Server tasks
- Reports
- Configuration

Evolveum | Projects | Role Catalog

Org. hierarchy

- Evolveum
  - Research and Development Division
    - Development Section
    - QA Section
    - Services Section
  - Business Division
    - Marketing Section
    - Accounting Section

Managers

Mgr. Igor Farinič (ifarinic)
CEO
✓ Enabled
Manager

Members

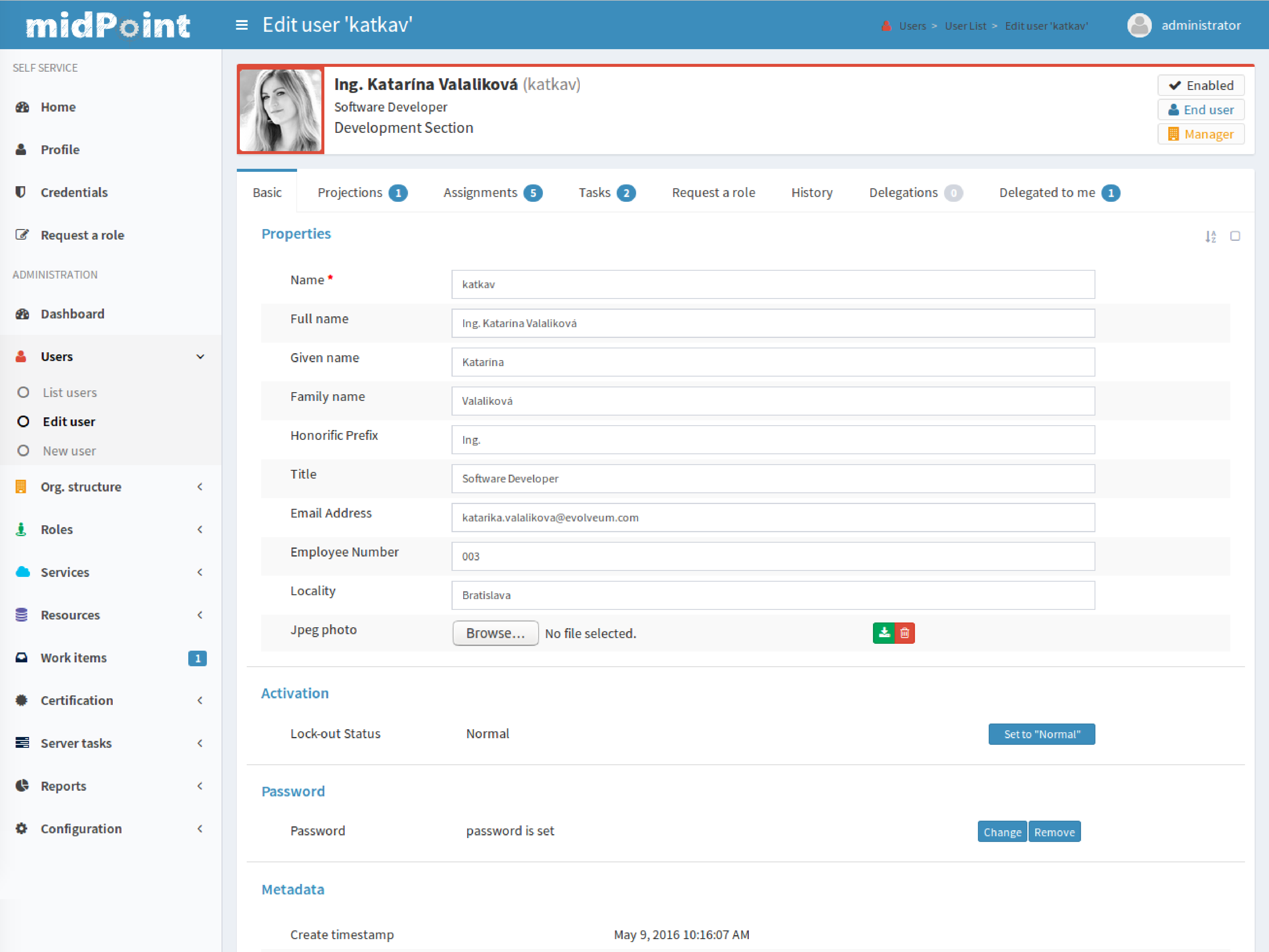
One level | Object

Name: All | More... | Advanced

	Type	Name	Fullname/Display name	Identifier/Description
☐		F1100	Research and Development Division	1100
☐		F1200	Business Division	1200
☐		ifarinic	Mgr. Igor Farinič	igor.farinic@evolveum.com

1 to 3 of 3 << < 1 > >>

Evolveum®


midPoint ≡ Edit user 'katkav'

Users > User List > Edit user 'katkav'

administrator

SELF SERVICE
- Home
- Profile
- Credentials
- Request a role

ADMINISTRATION
- Dashboard
- Users
  - List users
  - Edit user
  - New user
- Org. structure
- Roles
- Services
- Resources
- Work items 1
- Certification
- Server tasks
- Reports
- Configuration

Ing. Katarína Valaliková (katkav)
Software Developer
Development Section

Enabled
End user
Manager

Basic | Projections 1 | Assignments 5 | Tasks 2 | Request a role | History | Delegations 0 | Delegated to me 1

Properties

Name *	katkav
Full name	Ing. Katarina Valaliková
Given name	Katarina
Family name	Valaliková
Honorific Prefix	Ing.
Title	Software Developer
Email Address	katarika.valalikova@evolveum.com
Employee Number	003
Locality	Bratislava
Jpeg photo	Browse... No file selected.

Activation

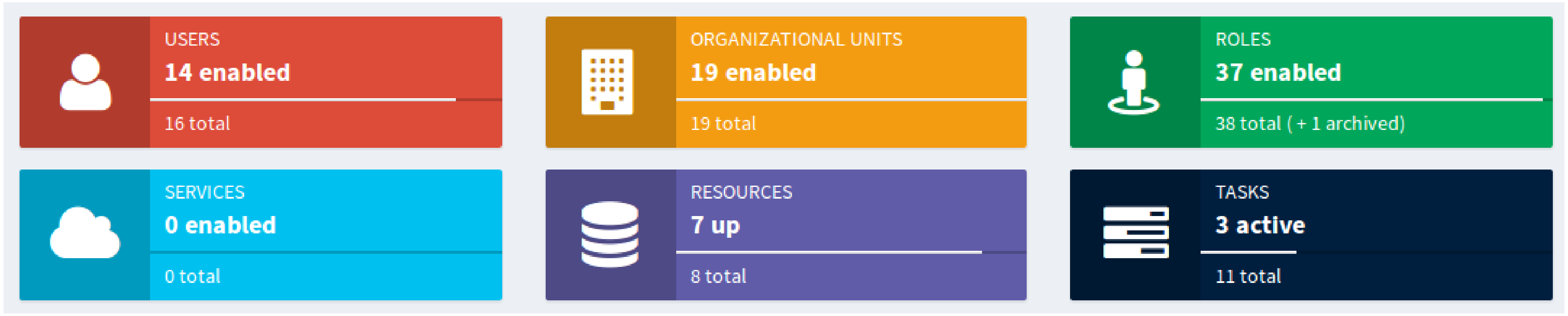
Lock-out Status: Normal — Set to "Normal"

Password

Password: password is set — Change | Remove

Metadata

Create timestamp: May 9, 2016 10:16:07 AM


USERS
14 enabled
16 total

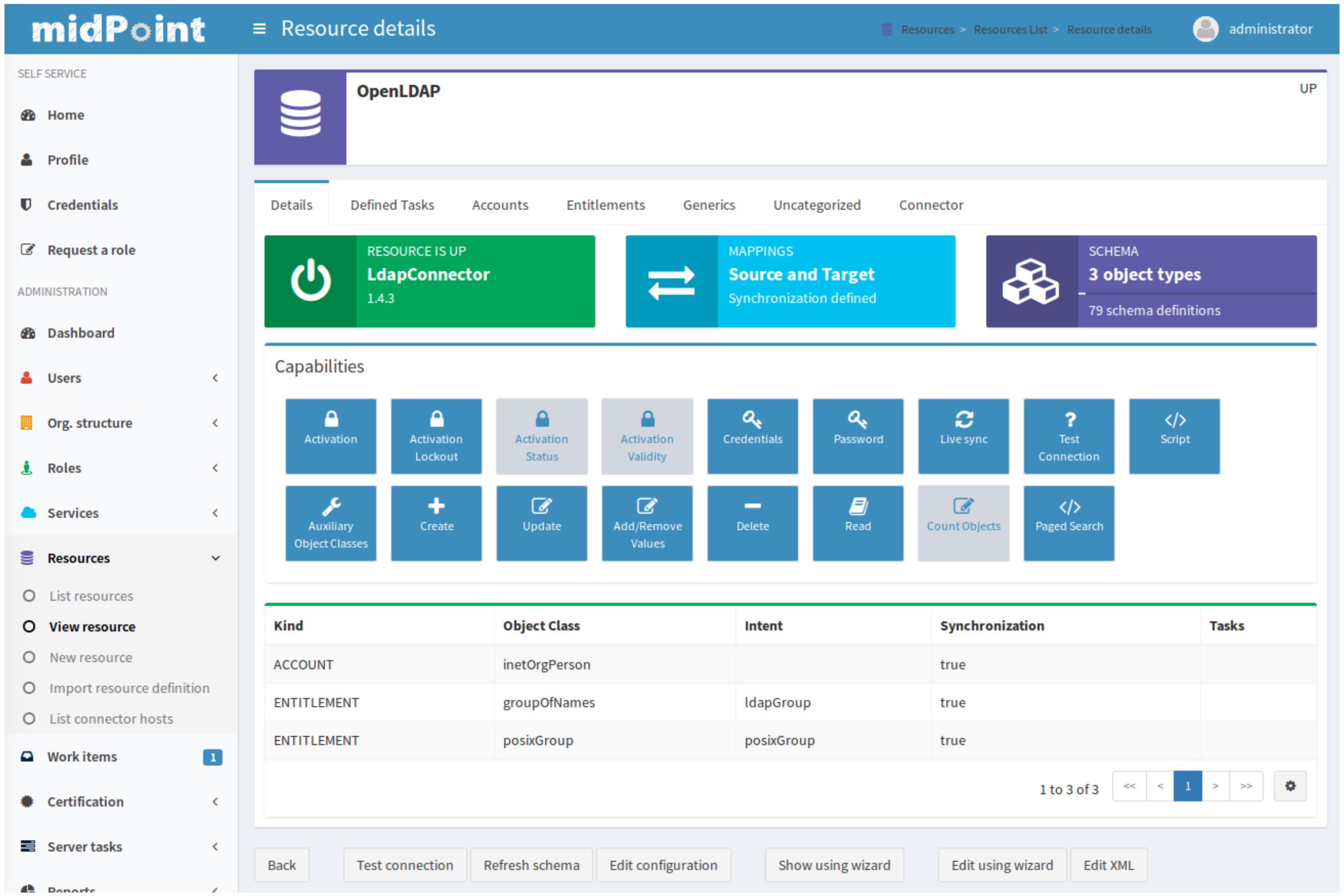
ORGANIZATIONAL UNITS
19 enabled
19 total

ROLES
37 enabled
38 total (+ 1 archived)

SERVICES
0 enabled
0 total

RESOURCES
7 up
8 total

TASKS
3 active
11 total



OpenLDAP
UP

RESOURCE IS UP
LdapConnector
1.4.3

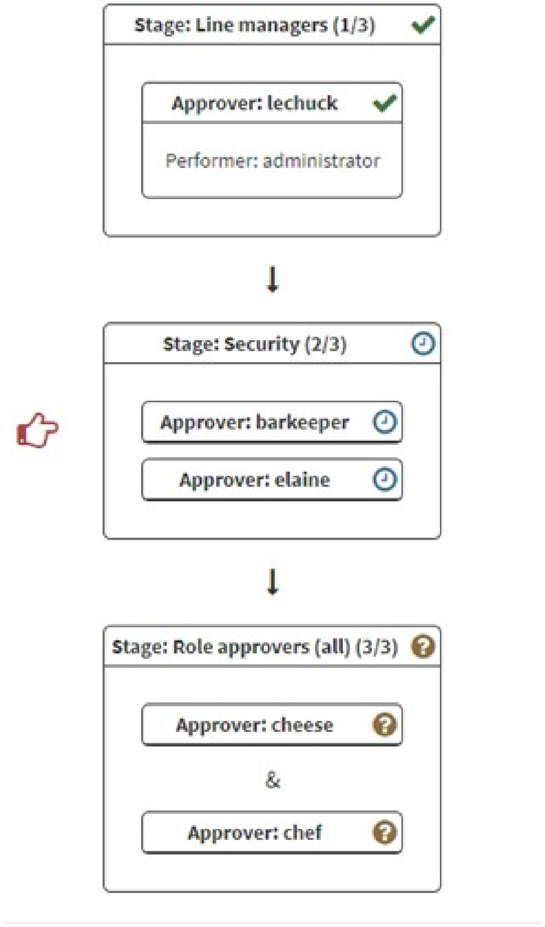
MAPPINGS
Source and Target
Synchronization defined

SCHEMA
3 object types
79 schema definitions

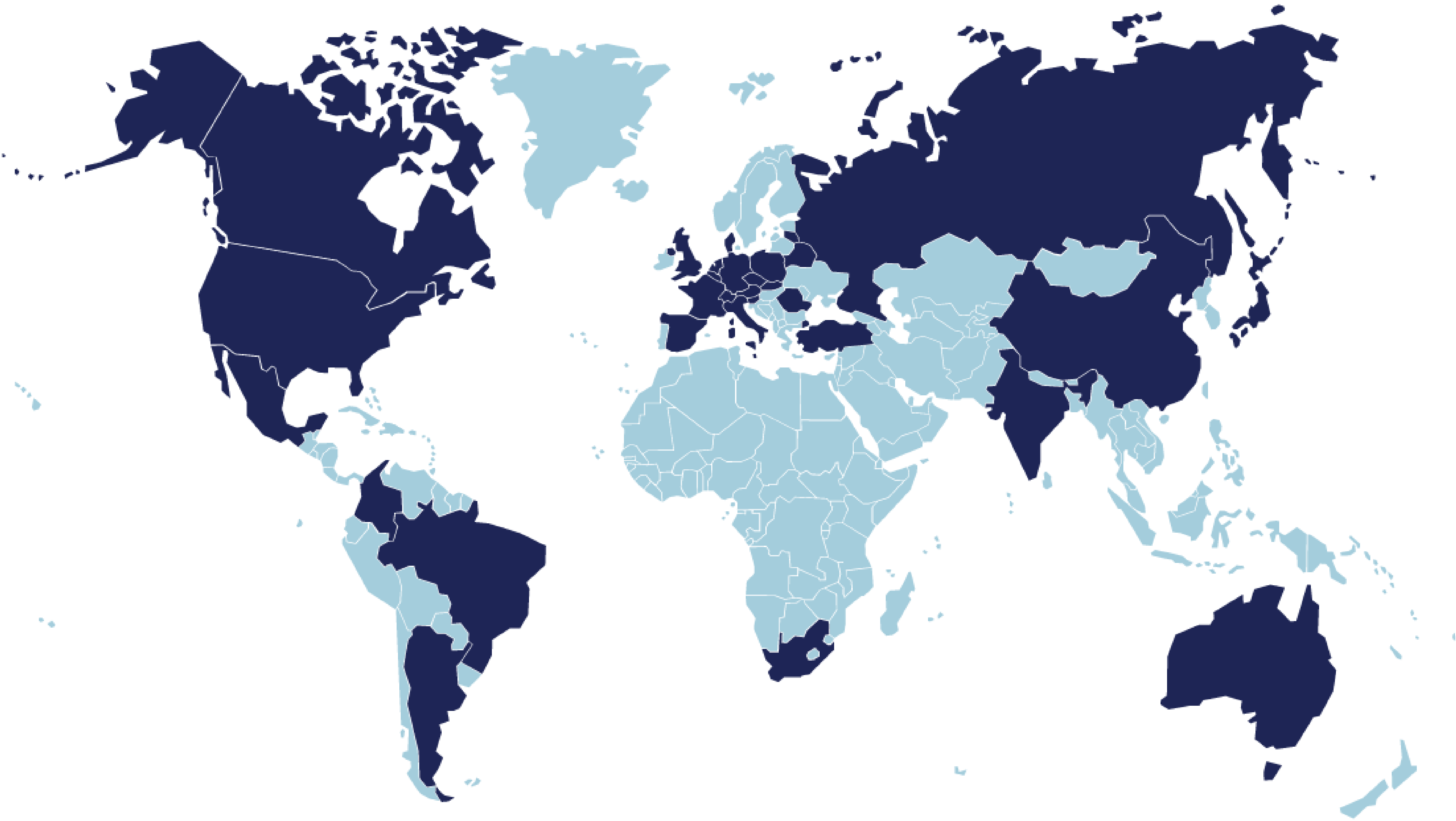
Capabilities

Kind	Object Class	Intent	Synchronization	Tasks
ACCOUNT	inetOrgPerson		true	
ENTITLEMENT	groupOfNames	ldapGroup	true	
ENTITLEMENT	posixGroup	posixGroup	true	

1 to 3 of 3

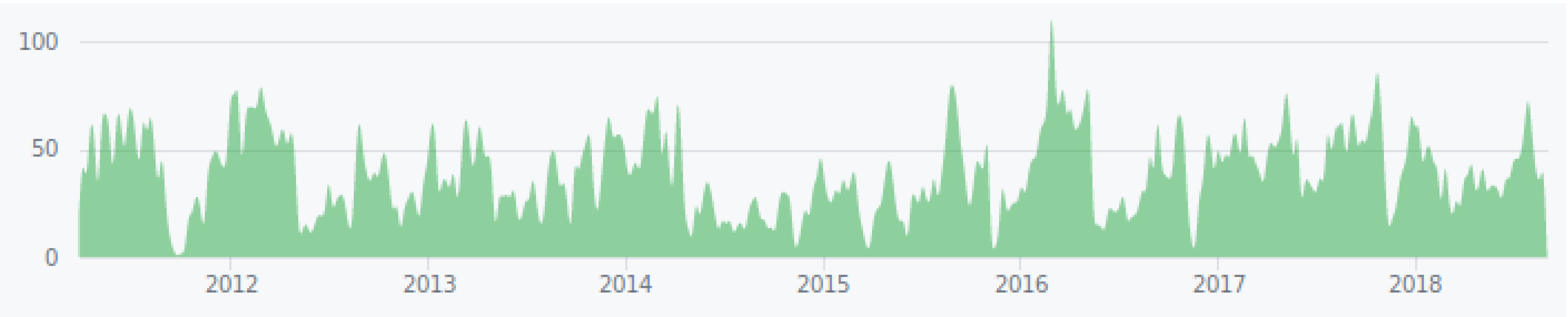


Evolveum®



MidPoint Development

- Everything is open source (see github)
- Evolutionary approach (iterative+incremental)
- At least 2 releases per year (26 releases)
- Team of 5 full-time developers (+contributors)
- High development activity (100-200 commits/month)



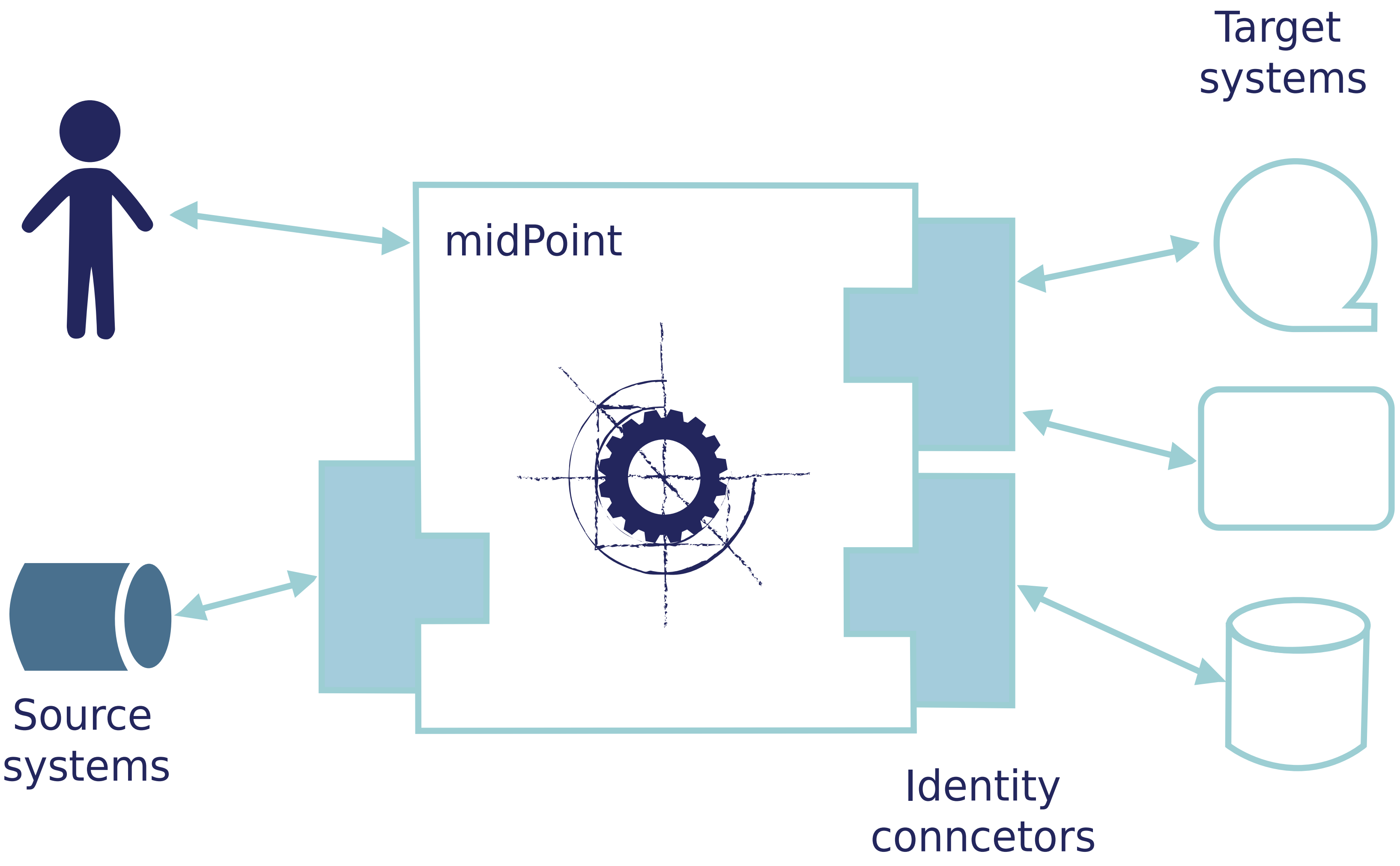
Let’s get back to technology ...

WARNING

HIC SUNT LEONES

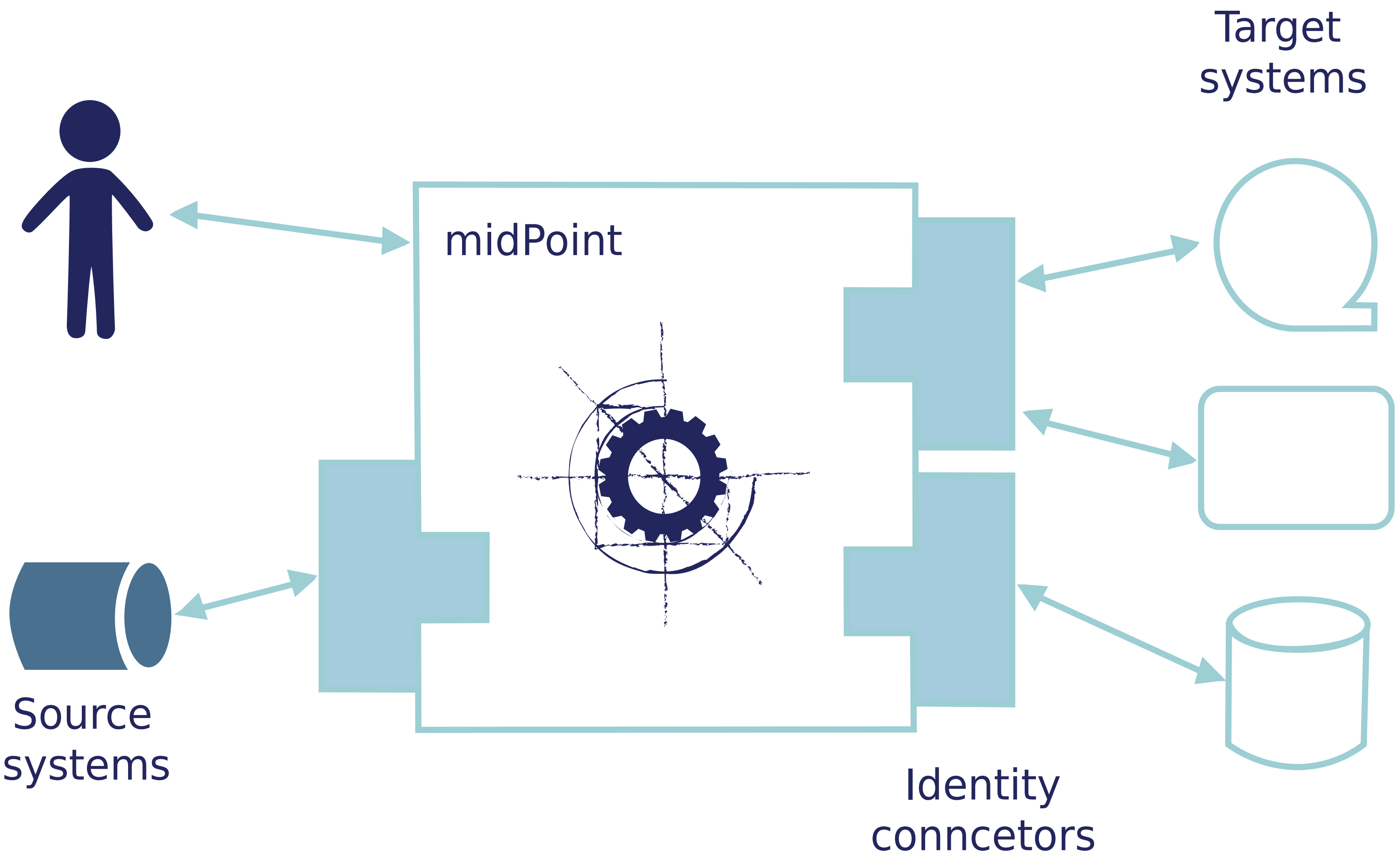
Controversial statements ahead!
Political correctness (very) limited.
Mental health hazards.
Dogmatic buzzword followers may be disturbed.

MidPoint Big Picture



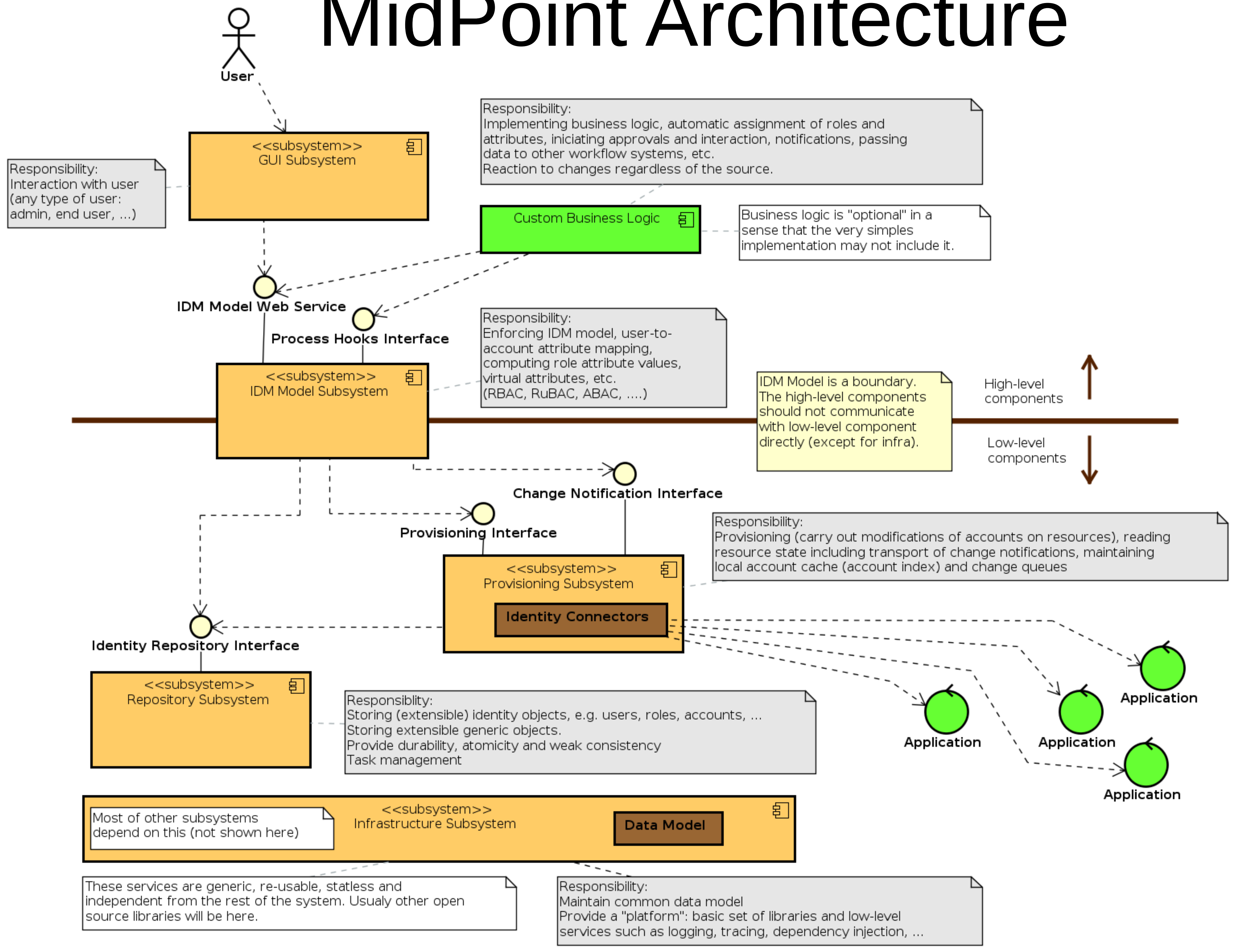
Evolveum

MidPoint Big Picture

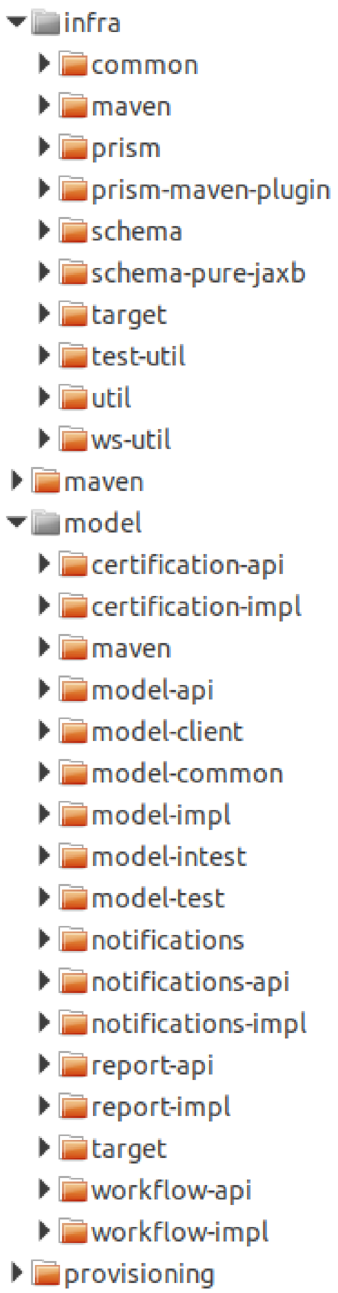


Monolith? Not really!

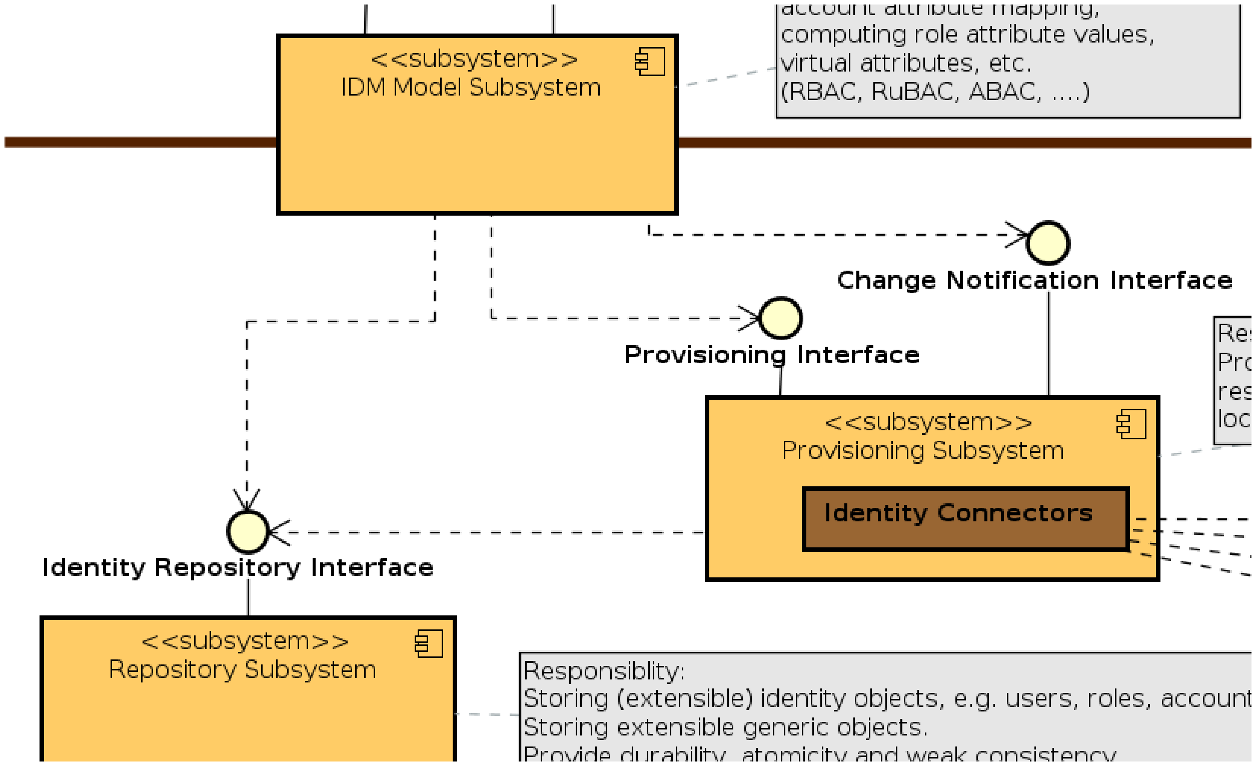
MidPoint Architecture


User
<<subsystem>>
GUI Subsystem
Responsibility:
Interaction with user
(any type of user:
admin, end user, ...)
Responsibility:
Implementing business logic, automatic assignment of roles and
attributes, iniciating approvals and interaction, notifications, passing
data to other workflow systems, etc.
Reaction to changes regardless of the source.
Custom Business Logic
Business logic is "optional" in a
sense that the very simples
implementation may not include it.
IDM Model Web Service
Process Hooks Interface
Responsibility:
Enforcing IDM model, user-to-
account attribute mapping,
computing role attribute values,
virtual attributes, etc.
(RBAC, RuBAC, ABAC, ...)
<<subsystem>>
IDM Model Subsystem
IDM Model is a boundary.
The high-level components
should not communicate
with low-level component
directly (except for infra).
High-level
components
Low-level
components
Change Notification Interface
Provisioning Interface
Responsibility:
Provisioning (carry out modifications of accounts on resources), reading
resource state including transport of change notifications, maintaining
local account cache (account index) and change queues
<<subsystem>>
Provisioning Subsystem
Identity Connectors
Identity Repository Interface
Application
Application
Application
Application
<<subsystem>>
Repository Subsystem
Responsiblity:
Storing (extensible) identity objects, e.g. users, roles, accounts, ...
Storing extensible generic objects.
Provide durability, atomicity and weak consistency
Task management
<<subsystem>>
Infrastructure Subsystem
Most of other subsystems
depend on this (not shown here)
Data Model
These services are generic, re-usable, statless and
independent from the rest of the system. Usualy other open
source libraries will be here.
Responsibility:
Maintain common data model
Provide a "platform": basic set of libraries and low-level
services such as logging, tracing, dependency injection, ...

Components, Source Code Structure



- infra
  - common
  - maven
  - prism
  - prism-maven-plugin
  - schema
  - schema-pure-jaxb
  - target
  - test-util
  - util
  - ws-util
- maven
- model
  - certification-api
  - certification-impl
  - maven
  - model-api
  - model-client
  - model-common
  - model-impl
  - model-intest
  - model-test
  - notifications
  - notifications-api
  - notifications-impl
  - report-api
  - report-impl
  - target
  - workflow-api
  - workflow-impl
- provisioning



account attribute mapping,
computing role attribute values,
virtual attributes, etc.
(RBAC, RuBAC, ABAC, ...)

Re
Pro
res
loc

Responsiblity:
Storing (extensible) identity objects, e.g. users, roles, account
Storing extensible generic objects.
Provide durability, atomicity and weak consistency

What you want to know but you are too afraid to ask

- Java? Really?
  - Really. And we use checked exceptions!
  - But no Java EE. We are not <u>that</u> crazy.
  - Compiler saves huge amount of time
    (you will see later: generated code)
  - Old language +1: libraries for everything
  - Old language -1: you need to avoid landmines
  - OpenJDK
  - Hindsight: Java is lesser evil

Dependencies (2010-2012)

- Spring
- Java Server Faces
- XML (DOM)
- JAX-B
- JAX-WS
- ESB (BPEL)
- Activiti BPM (BPMN.2)
- Jasper Reports
- Hibernate

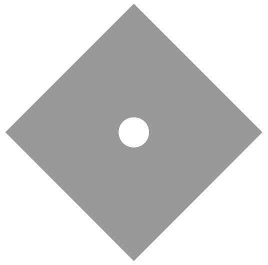
Dependencies (2018)

- Spring + Spring Boot
- ~~Java Server Faces~~ Apache Wicket
- XML (DOM) + JSON + YAML
- ~~JAX-B~~ : (almost) replaced
- ~~JAX-WS~~ : not used much any more
- ~~ESB (BPEL)~~ : replaced before midPoint started
- ~~Activiti BPM (BPMN.2)~~ : being replaced right now
- Jasper Reports : not that useful, will it survive?
- Hibernate : may be replaced later on

Dependencies : Lessons Learned

- Faster start of the project
- Do not reinvent the wheel

  … unless the wheel is in fact a square



- Do not depend on dependencies too much
- Understand how they work – and why they fail
- Have a “Plan B” to replace them later on

Architecture?

“REST”, Microservices, Web frameworks, …

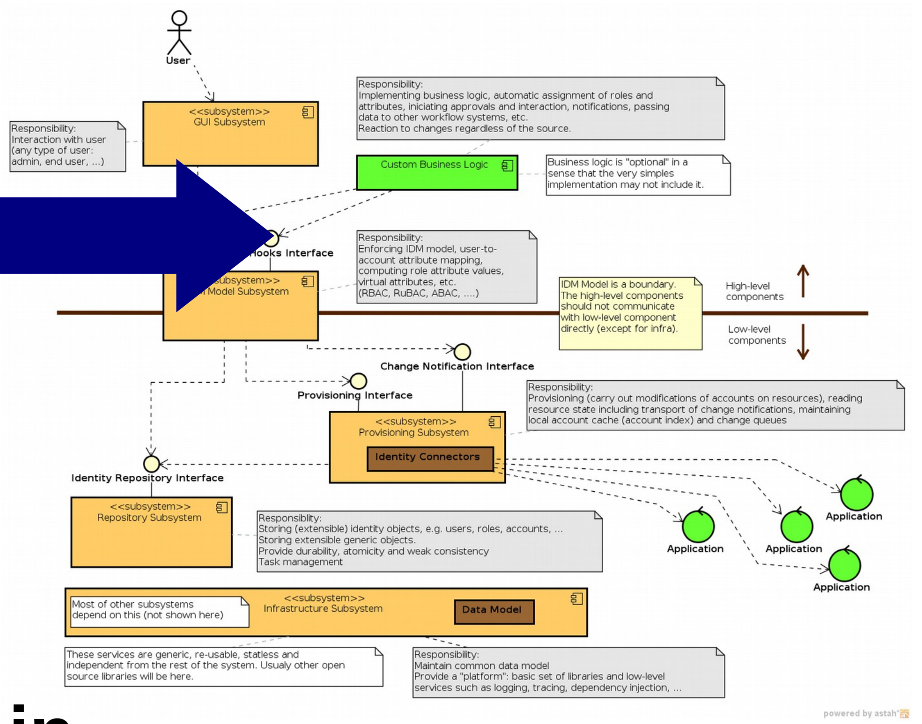
That’s not architecture!

Architecture!

“REST”, Microservices, Web frameworks, ...

That’s not architecture!

This is architecture


User
<<subsystem>> GUI Subsystem
Custom Business Logic
Hooks Interface
<<subsystem>> Model Subsystem
High-level components
Low-level components
Change Notification Interface
Provisioning Interface
<<subsystem>> Provisioning Subsystem
Identity Connectors
Identity Repository Interface
<<subsystem>> Repository Subsystem
Application
Application
Application
Application
<<subsystem>> Infrastructure Subsystem
Data Model
powered by astah*

Responsibility:
Interaction with user
(any type of user:
admin, end user, ...)

Responsibility:
Implementing business logic, automatic assignment of roles and attributes, iniciating approvals and interaction, notifications, passing data to other workflow systems, etc.
Reaction to changes regardless of the source.

Business logic is "optional" in a sense that the very simples implementation may not include it.

Responsibility:
Enforcing IDM model, user-to-account attribute mapping, computing role attribute values, virtual attributes, etc.
(RBAC, RuBAC, ABAC, ...)

IDM Model is a boundary. The high-level components should not communicate with low-level component directly (except for infra).

Responsibility:
Provisioning (carry out modifications of accounts on resources), reading resource state including transport of change notifications, maintaining local account cache (account index) and change queues

Responsiblity:
Storing (extensible) identity objects, e.g. users, roles, accounts, ...
Storing extensible generic objects.
Provide durability, atomicity and weak consistency
Task management

Most of other subsystems depend on this (not shown here)

These services are generic, re-usable, statless and independent from the rest of the system. Usualy other open source libraries will be here.

Responsibility:
Maintain common data model
Provide a "platform": basic set of libraries and low-level services such as logging, tracing, dependency injection, ...

Components, subsystems, interfaces, modules, separation of concerns

You really should pay attention in software engineering classes.

Evolveum®

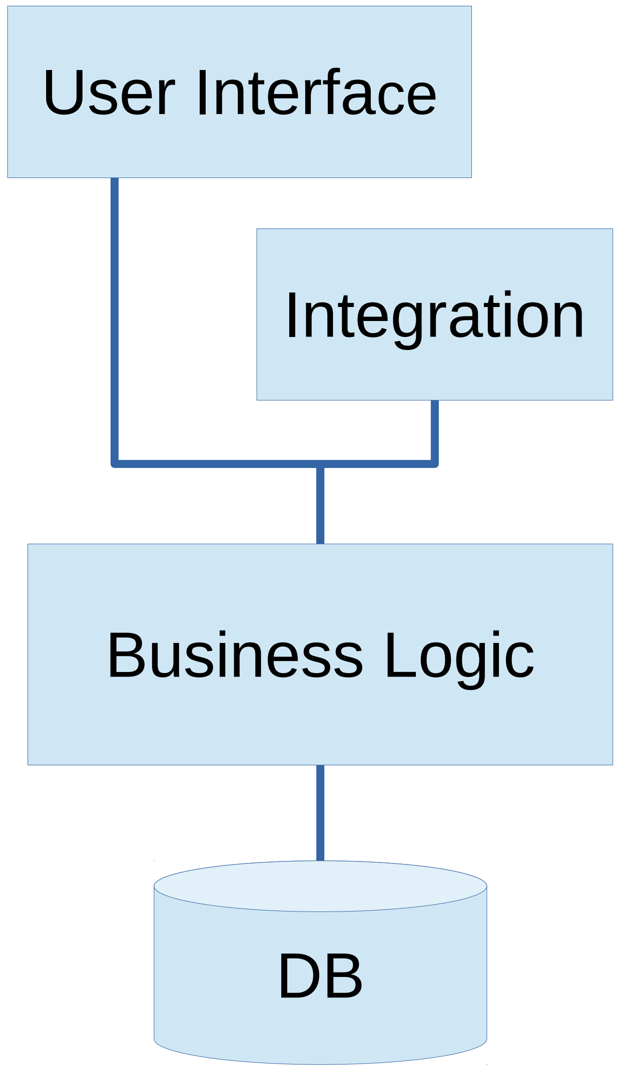
How Does This *Architecture* Help With Maintainability?

- Component encapsulation (cohesion, coupling)
  - Limited impact of changes
  - … and changes **will** happen
- Interfaces = abstraction
  - Controlling how far changes can “spread”
  - Compatibility
- Modularity
  - Changing components (implementation) without impacting other components

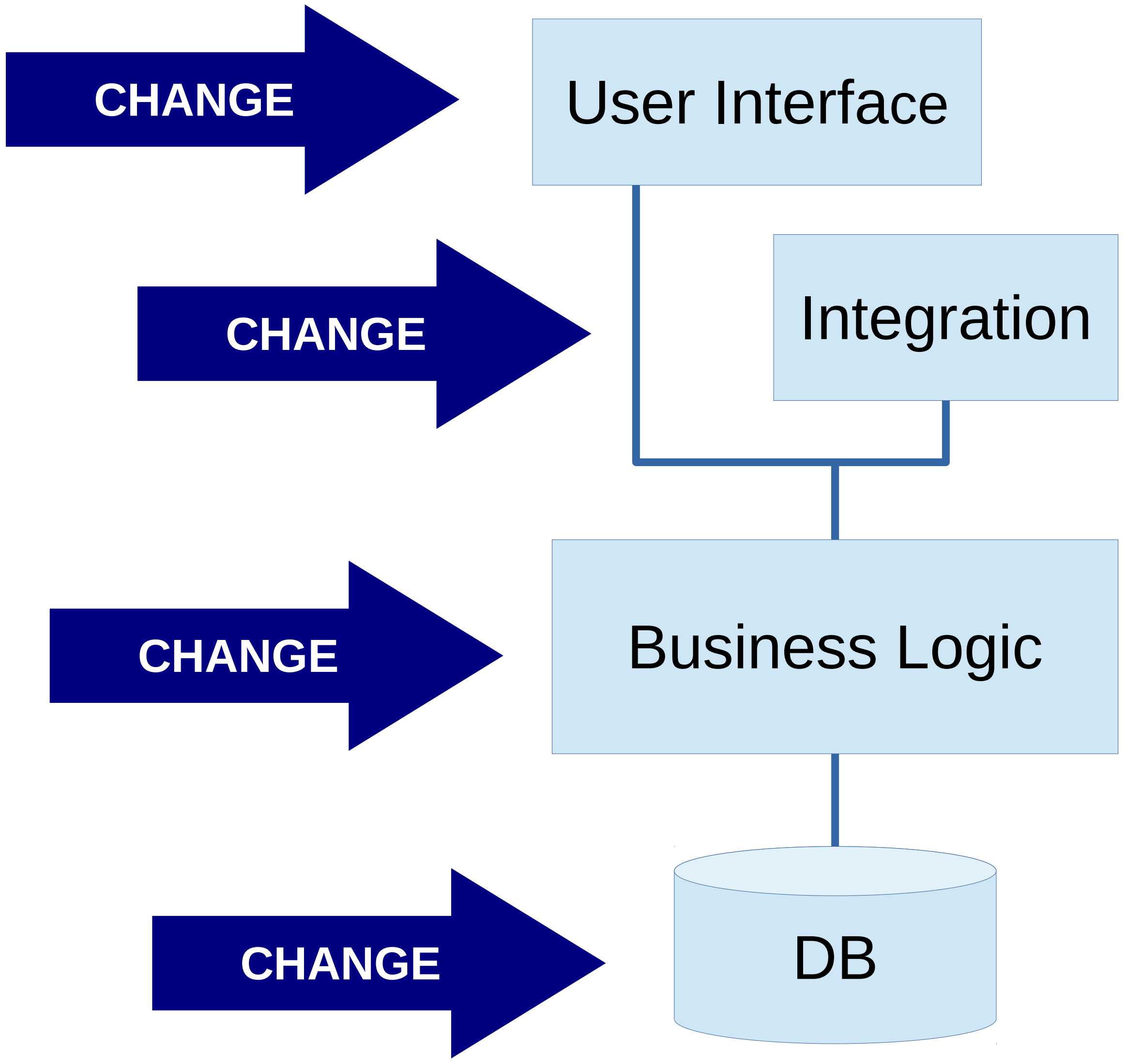
Data Model

- Extremely important
- As important as architecture
- Cross-cutting concern
- Performance, scalability, evolvability, …
- Changes often – especially at the beginning
- Evolution - compatibility
- Experimental features

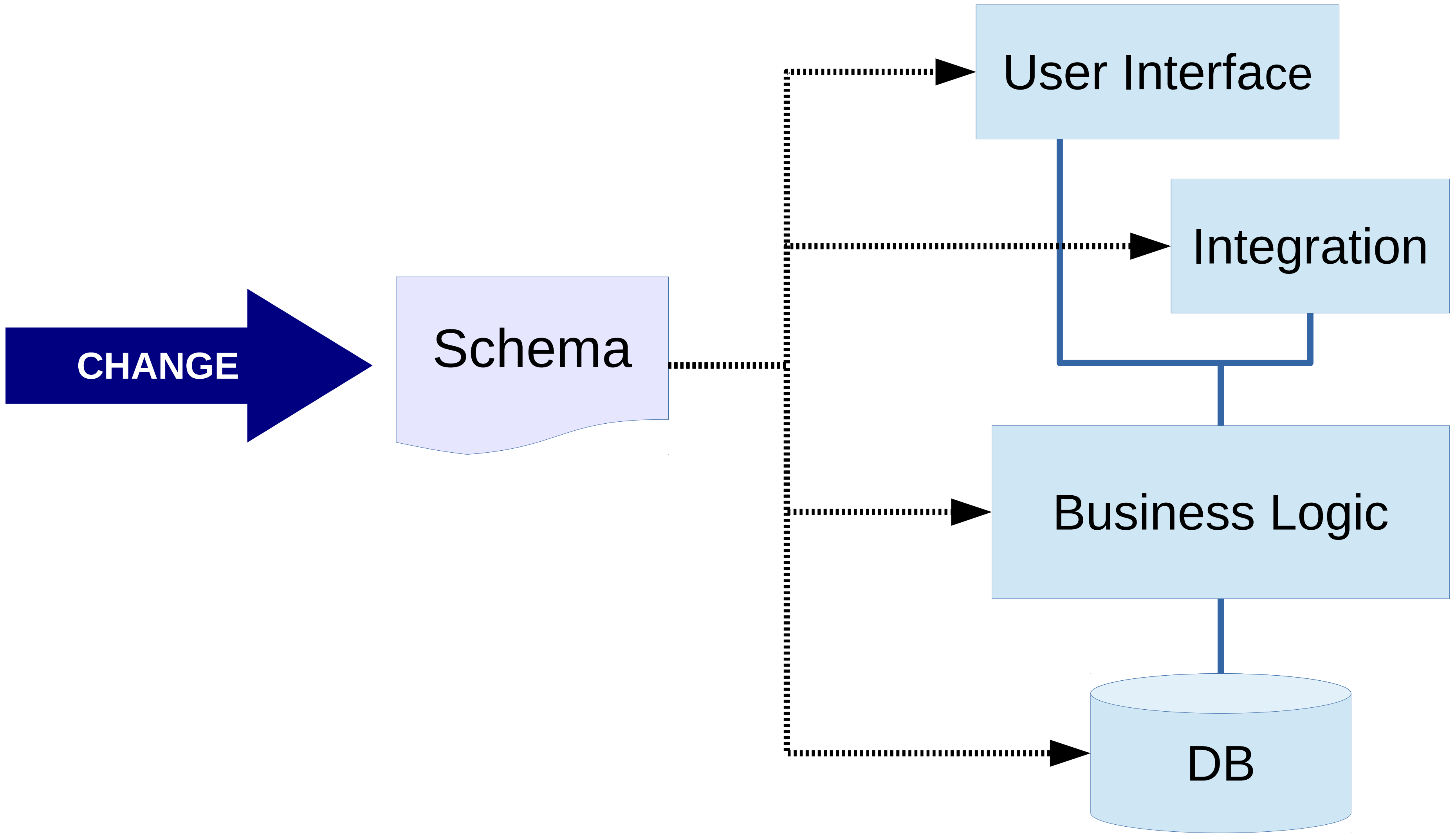
Data Model



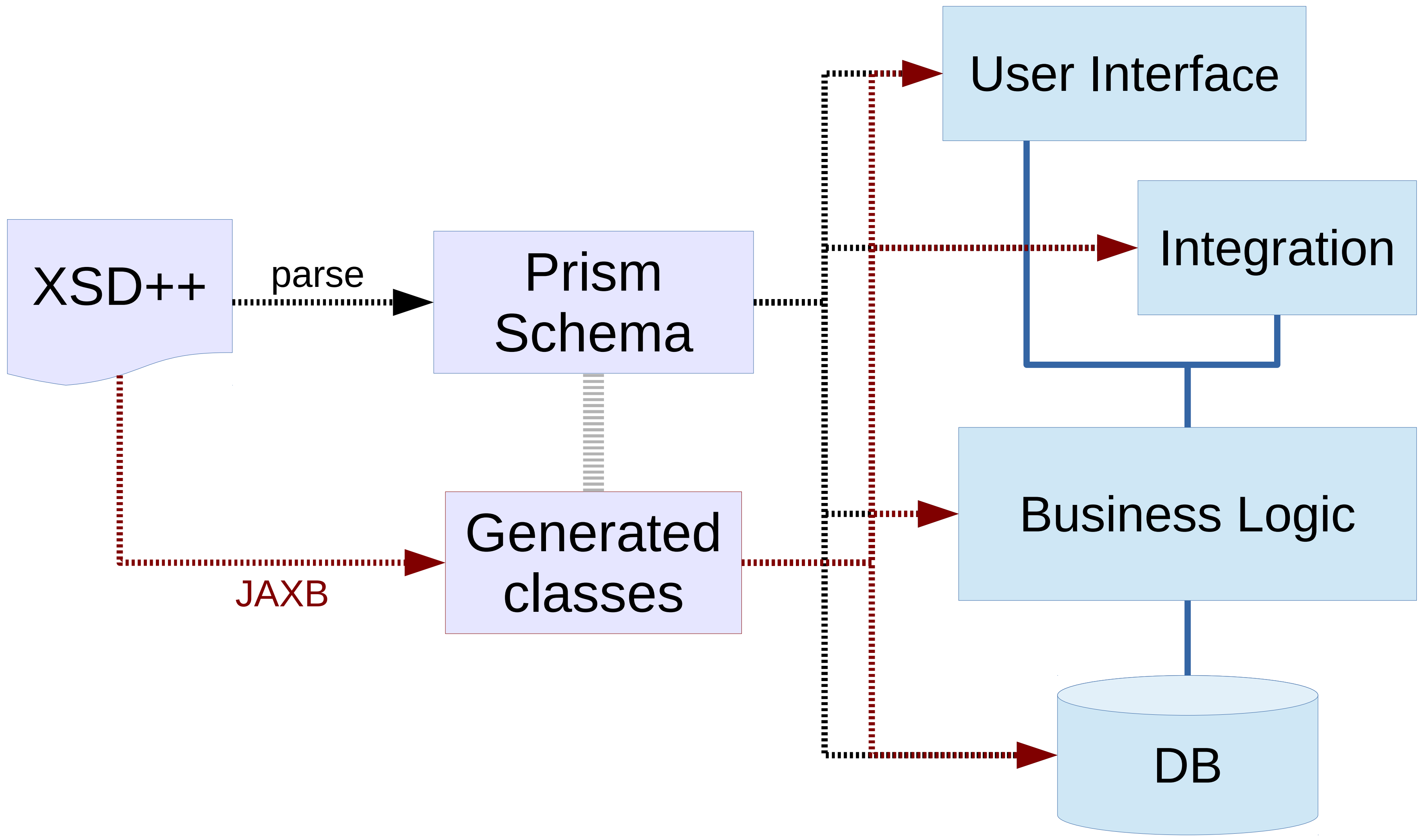
Data Model Change



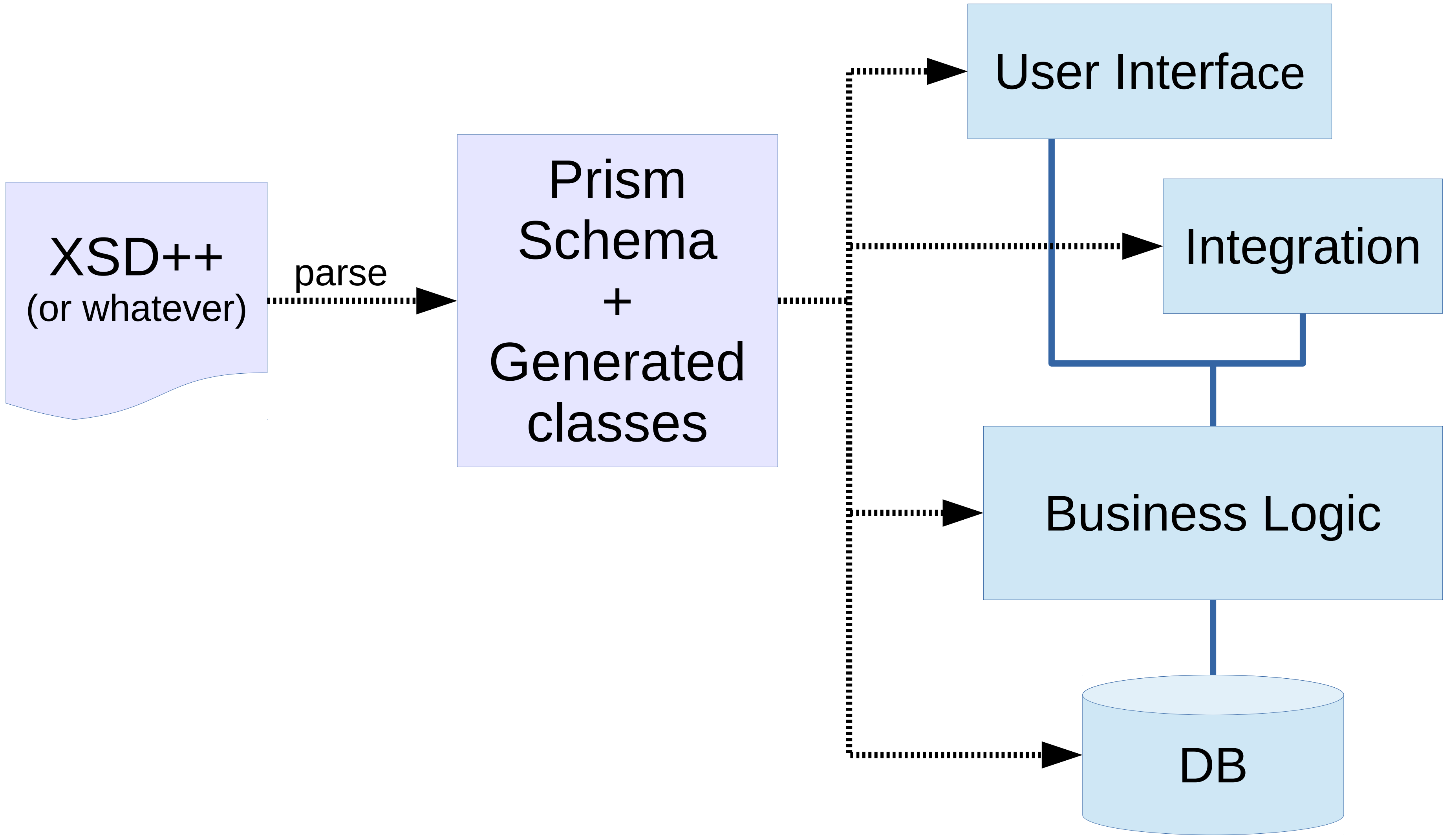
Data Model : Schema



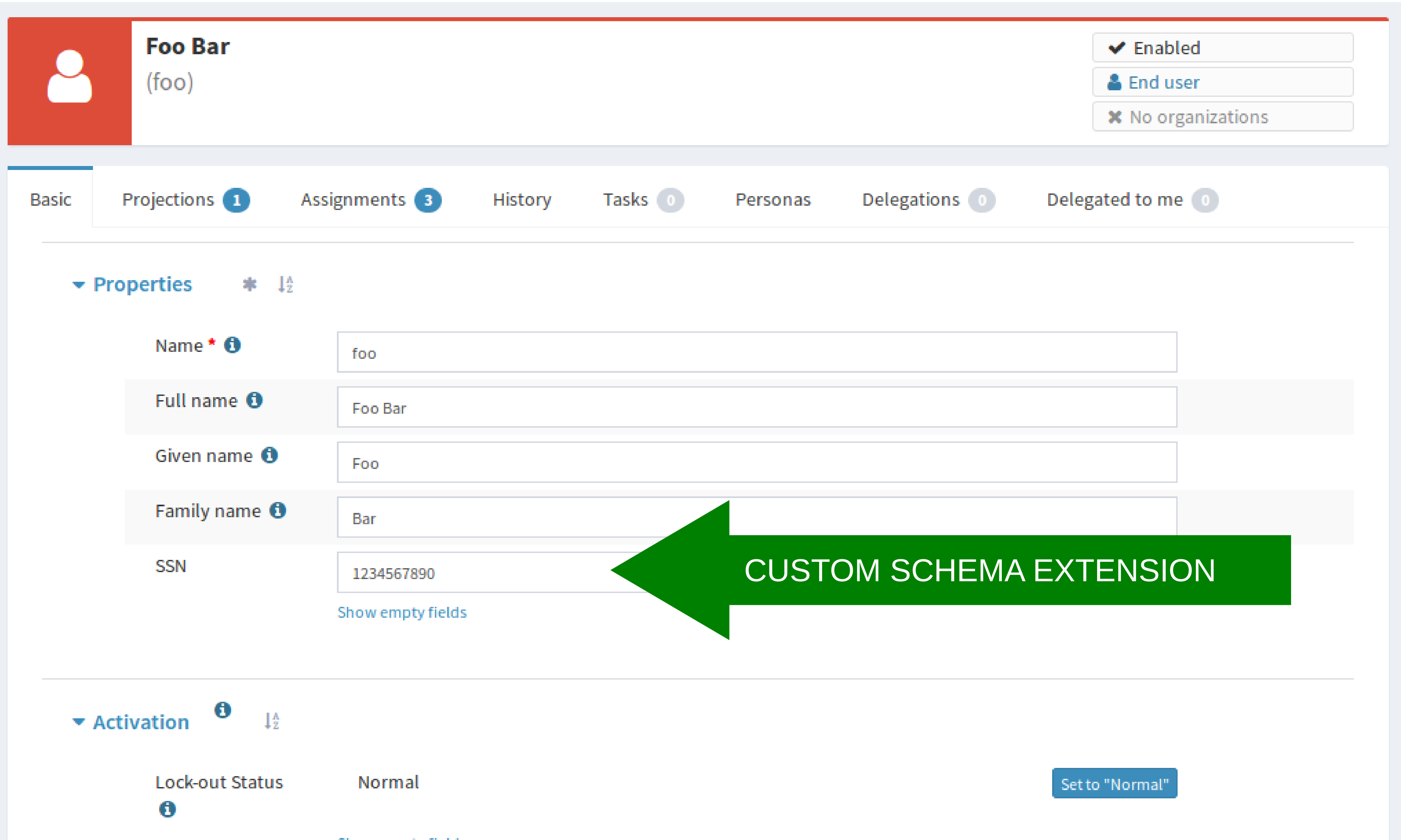
MidPoint : Prism Schema (now)



MidPoint : Prism Schema (future)



MidPoint : Prism Schema in UI



Foo Bar
(foo)

✓ Enabled
End user
✖ No organizations

Basic | Projections 1 | Assignments 3 | History | Tasks 0 | Personas | Delegations 0 | Delegated to me 0

Properties

Name *	foo
Full name	Foo Bar
Given name	Foo
Family name	Bar
SSN	1234567890

Show empty fields

CUSTOM SCHEMA EXTENSION

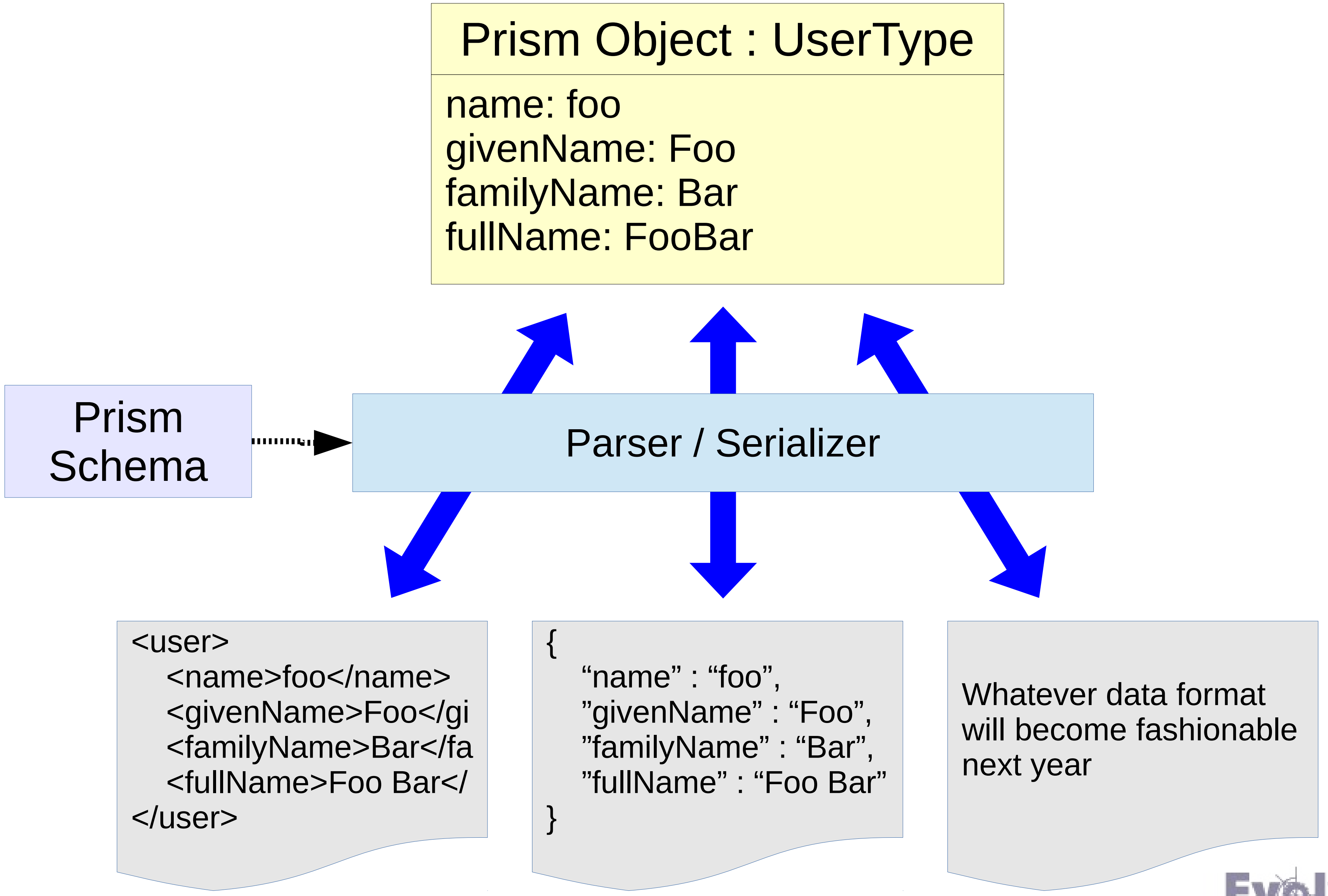
Activation

Lock-out Status	Normal	Set to "Normal"

Questions You Surely Want To Ask

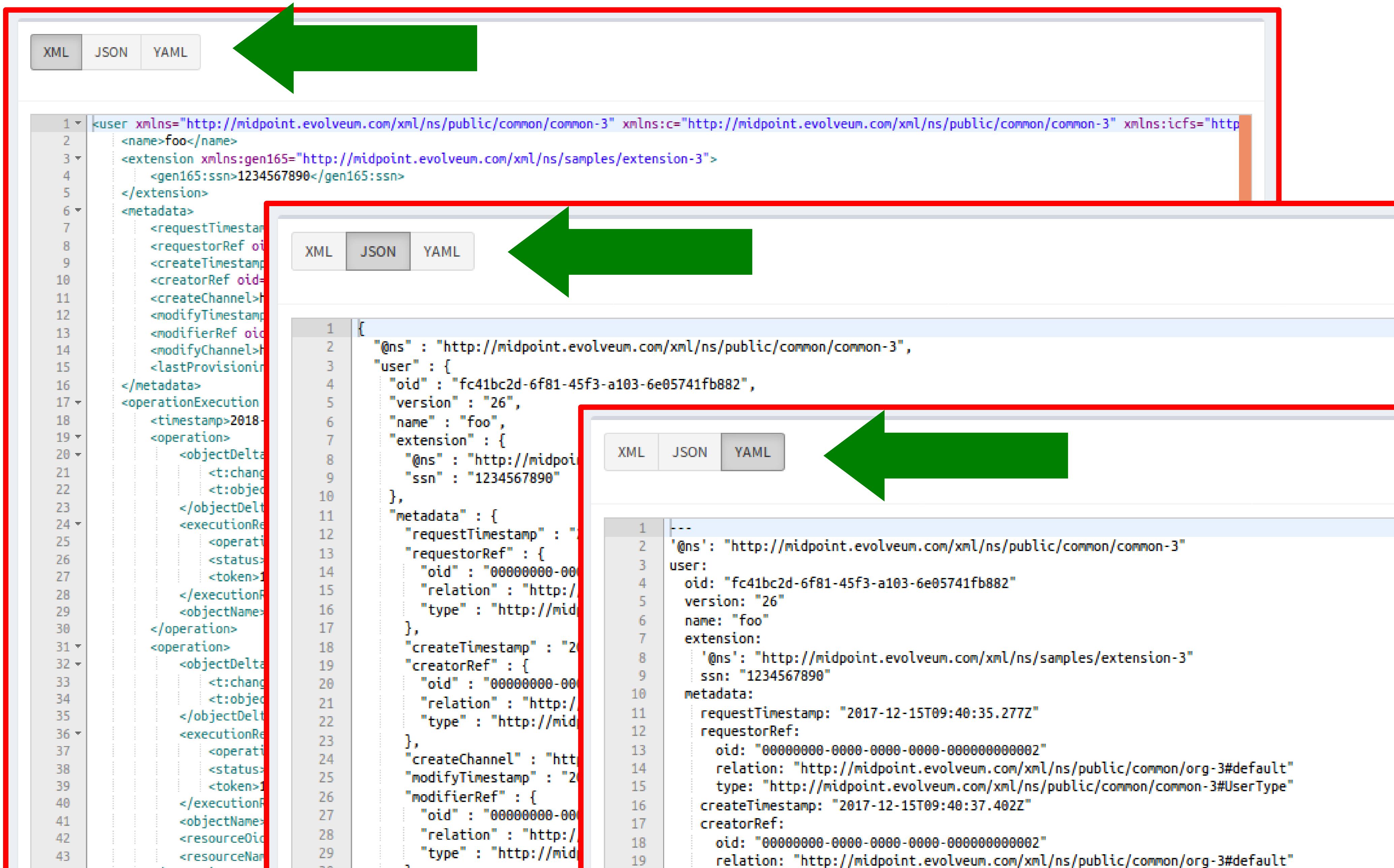
- Why XML and XSD? That's not cool any more!
  - Because midPoint started in 2011
  - Because JSON is not much better than XML … and YAML is even worse
  - Because JSON Schema and others are equally bad
  - **New** technology does not mean **better** technology (except when it does)
  - Anyway, we are reaching limits of XML/XSD likely change in the future

XML, JSON, YAML and Friends



Evolveum

XML, JSON, YAML and Friends



```
<user xmlns="http://midpoint.evolveum.com/xml/ns/public/common/common-3" xmlns:c="http://midpoint.evolveum.com/xml/ns/public/common/common-3" xmlns:icfs="http
    <name>foo</name>
    <extension xmlns:gen165="http://midpoint.evolveum.com/xml/ns/samples/extension-3">
        <gen165:ssn>1234567890</gen165:ssn>
    </extension>
    <metadata>
        <requestTimestam
        <requestorRef oi
        <createTimestamp
        <creatorRef oid=
        <createChannel>h
        <modifyTimestamp
        <modifierRef oid
        <modifyChannel>h
        <lastProvisionin
    </metadata>
    <operationExecution
        <timestamp>2018-
        <operation>
            <objectDelta
                <t:chang
                <t:objec
            </objectDelt
            <executionRe
                <operati
                <status>
                <token>1
            </executionR
            <objectName>
        </operation>
        <operation>
            <objectDelta
                <t:chang
                <t:objec
            </objectDelt
            <executionRe
                <operati
                <status>
                <token>1
            </executionR
            <objectName>
            <resourceOic
            <resourceNam
```

```
{
  "@ns" : "http://midpoint.evolveum.com/xml/ns/public/common/common-3",
  "user" : {
    "oid" : "fc41bc2d-6f81-45f3-a103-6e05741fb882",
    "version" : "26",
    "name" : "foo",
    "extension" : {
      "@ns" : "http://midpoi
      "ssn" : "1234567890"
    },
    "metadata" : {
      "requestTimestamp" : "
      "requestorRef" : {
        "oid" : "00000000-00
        "relation" : "http:/
        "type" : "http://mid
      },
      "createTimestamp" : "2
      "creatorRef" : {
        "oid" : "00000000-00
        "relation" : "http:/
        "type" : "http://mid
      },
      "createChannel" : "htt
      "modifyTimestamp" : "2
      "modifierRef" : {
        "oid" : "00000000-00
        "relation" : "http:/
        "type" : "http://mid
```

```
---
'@ns': "http://midpoint.evolveum.com/xml/ns/public/common/common-3"
user:
  oid: "fc41bc2d-6f81-45f3-a103-6e05741fb882"
  version: "26"
  name: "foo"
  extension:
    '@ns': "http://midpoint.evolveum.com/xml/ns/samples/extension-3"
    ssn: "1234567890"
  metadata:
    requestTimestamp: "2017-12-15T09:40:35.277Z"
    requestorRef:
      oid: "00000000-0000-0000-0000-000000000002"
      relation: "http://midpoint.evolveum.com/xml/ns/public/common/org-3#default"
      type: "http://midpoint.evolveum.com/xml/ns/public/common/common-3#UserType"
    createTimestamp: "2017-12-15T09:40:37.402Z"
    creatorRef:
      oid: "00000000-0000-0000-0000-000000000002"
      relation: "http://midpoint.evolveum.com/xml/ns/public/common/org-3#default"
```

How Does This *Schema* Thing Help With Maintainability?

- Evolution of data model is easy
  - Change schema → everything else adapts
  - Easy to add new features
- Compatibility control
  - Incompatible change → compilation goes

- Easy adaptation to environment
  - If some FooML becomes fashionable next year, we can easily support that

RESTful Interface

REST (almost)

RPC

```
http://…/rest/users
http://…/rest/users/02c15378-c48b-11e7-b010-1ff8606bae23
http://…/rest/tasks/c68d7770-...-9bec1fc3b57c/suspend
http://…/rest/notifyChange
```

- “REST” part and RPC part (and some overlap)
- Full schema support: XML, JSON, YAML
- Big problem of REST: modifications

  … but we do not worry, we have deltas
- SOAP to REST in five easy steps

See my lecture on midPoint REST for all the details (Nov 2017)

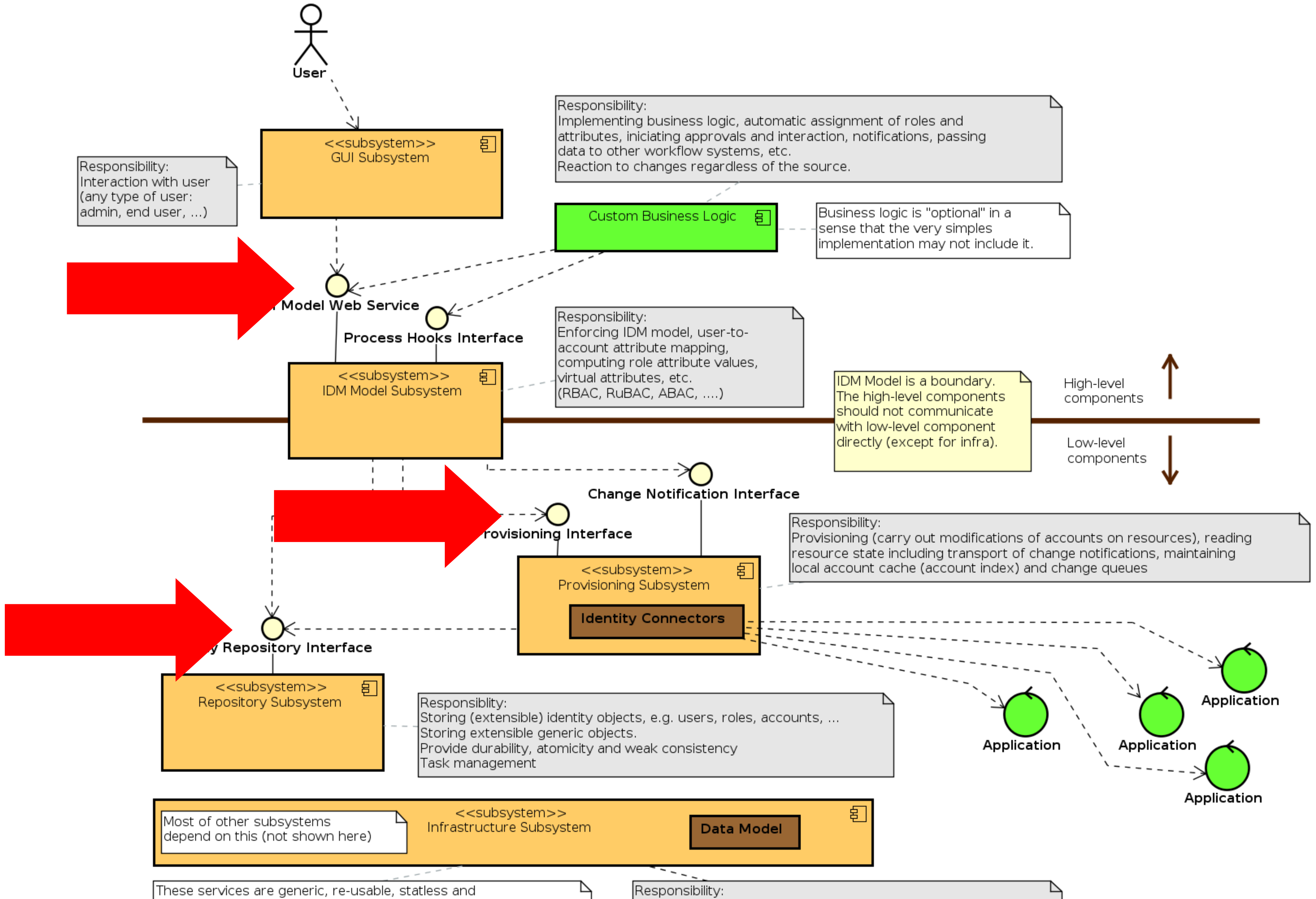
How Does *REST API* Help With Maintainability?

- It’s not really *REST API*, it is a REST-inspired interface … but don’t let me get started on this
- Third-party extensions of the system
  - You cannot predict all possible use-cases
  - Other people will add functionality, integrate, …
- It is an interface
  - Implementation may be different, but RESTful interface will stay compatible
  - Isolate outside of the system and inside of the system

Testing

- Automated **integration testing**
  - Thousands of test cases
  - Still based on unit test framework (TestNG)
  - Embed what you can (DB, LDAP server, ...)
- Not that much **unit tests**
  - Are you crazy? *Yes, we are … I mean: No!*
  - Remember: code generated from schema + compiler
  - Unit test maintenance is <u>very</u> expensive
- End-to-end tests – in progress
- Test-Driven Bugfixing (TDB)

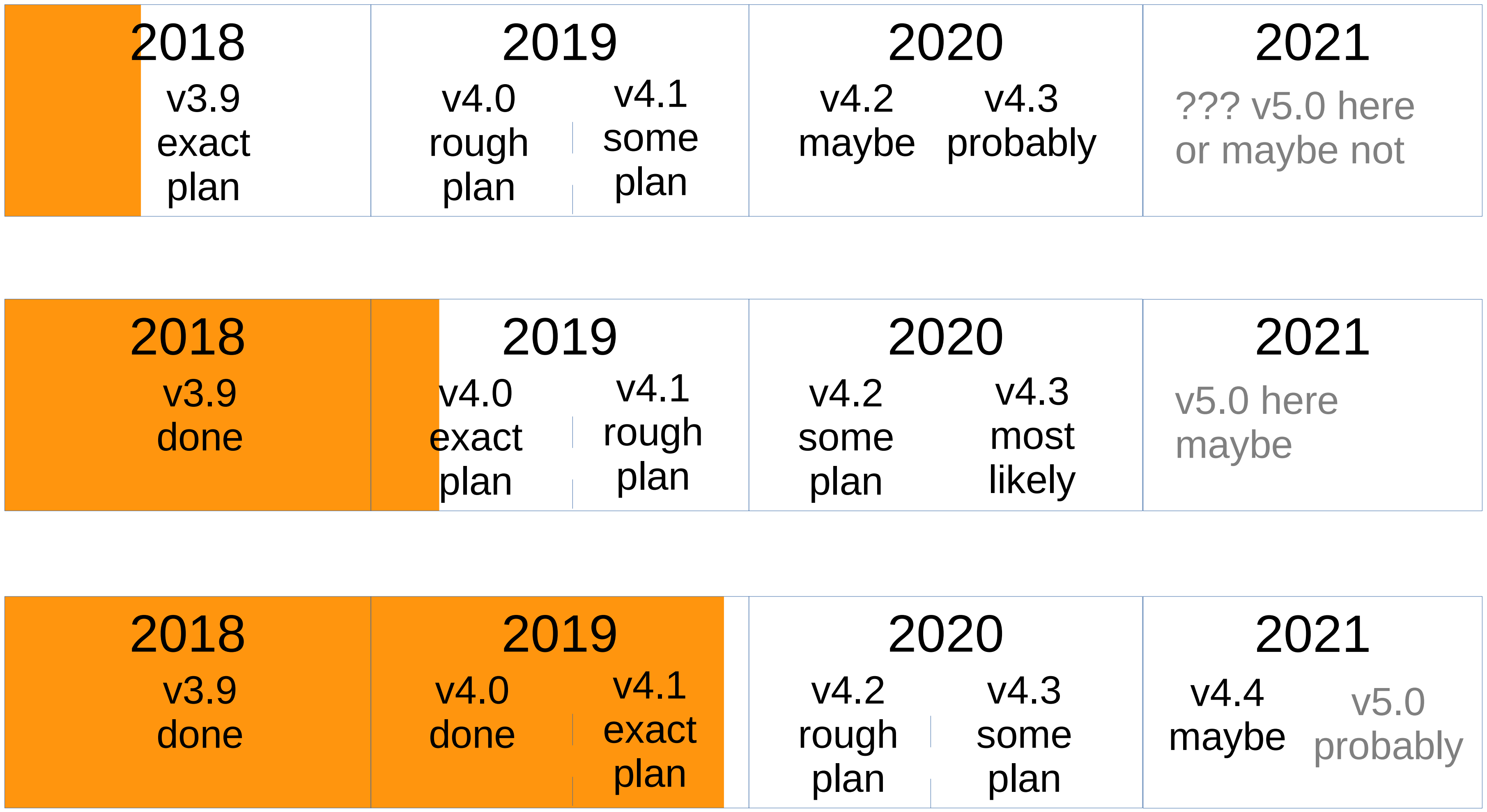
Designed For (Integration) Testability


User
<<subsystem>>
GUI Subsystem
Responsibility:
Interaction with user
(any type of user:
admin, end user, ...)
Responsibility:
Implementing business logic, automatic assignment of roles and
attributes, iniciating approvals and interaction, notifications, passing
data to other workflow systems, etc.
Reaction to changes regardless of the source.
Custom Business Logic
Business logic is "optional" in a
sense that the very simples
implementation may not include it.
Model Web Service
Process Hooks Interface
Responsibility:
Enforcing IDM model, user-to-
account attribute mapping,
computing role attribute values,
virtual attributes, etc.
(RBAC, RuBAC, ABAC, ...)
<<subsystem>>
IDM Model Subsystem
IDM Model is a boundary.
The high-level components
should not communicate
with low-level component
directly (except for infra).
High-level
components
Low-level
components
Change Notification Interface
rovisioning Interface
Responsibility:
Provisioning (carry out modifications of accounts on resources), reading
resource state including transport of change notifications, maintaining
local account cache (account index) and change queues
<<subsystem>>
Provisioning Subsystem
Identity Connectors
y Repository Interface
Application
Application
Application
Application
<<subsystem>>
Repository Subsystem
Responsiblity:
Storing (extensible) identity objects, e.g. users, roles, accounts, ...
Storing extensible generic objects.
Provide durability, atomicity and weak consistency
Task management
Most of other subsystems
depend on this (not shown here)
<<subsystem>>
Infrastructure Subsystem
Data Model
These services are generic, re-usable, statless and
Responsibility:

How Can *Testing* Ever Help With Maintainability?

- Automated testing is absolutely crucial!
- Continuous “Integration”
  - You cannot retest everything manually after each commit. But Jenkins can!
- You cannot do serious refactoring without good automated tests
  - If you cannot refactor you will drown in your own garbage (much sooner than you think)

Rolling-Wave Approach



Evolveum®

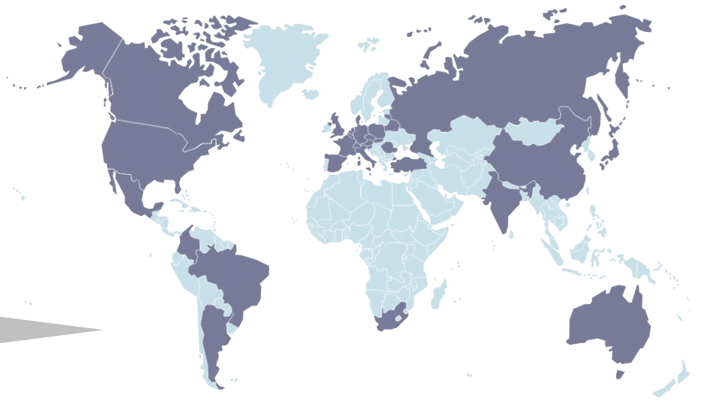
Rolling-Wave Approach

- Rolling-wave planning: obvious and intuitive
- Rolling-wave approach applied to everything:
  - architecture, schema, features, release scope
- Create architecture that can survive decades
  - But do NOT implement everything
  - Implement only what you need now
- Design 1-3 years ahead
  - But do NOT implement what you don’t need now
  - Data model (schema), DB model, interfaces
- Implement only what you need

How Can Such *Evolutionary Approach* Help With Maintainability?

- Design early → less rework later
  - It is easy to change the design any time before the implementation starts
- Design only, do not implement!
  - If you implement early, you will have too much to maintain and rework
- Do not be afraid to change the plans
  - Only bad plan cannot be changed
- We are <u>Evolve</u>um after all!

Questions you wanted to ask at the beginning



- Self-funded? And still alive?
  - Alive and well.
  - Bootstrapped (FFF). No venture capital.
  - Beginnings were hard. Very hard.
  - Persistence pays off.
- Business model?
  - Subscription: support + new feature development
  - Trainings, PoCs, Architecture reviews
  - Professional services, projects (minimal) → partners

Join the Team

- Java developers, IDM engineers, …

- Košice, Bratislava

  ... or anywhere (remote work)

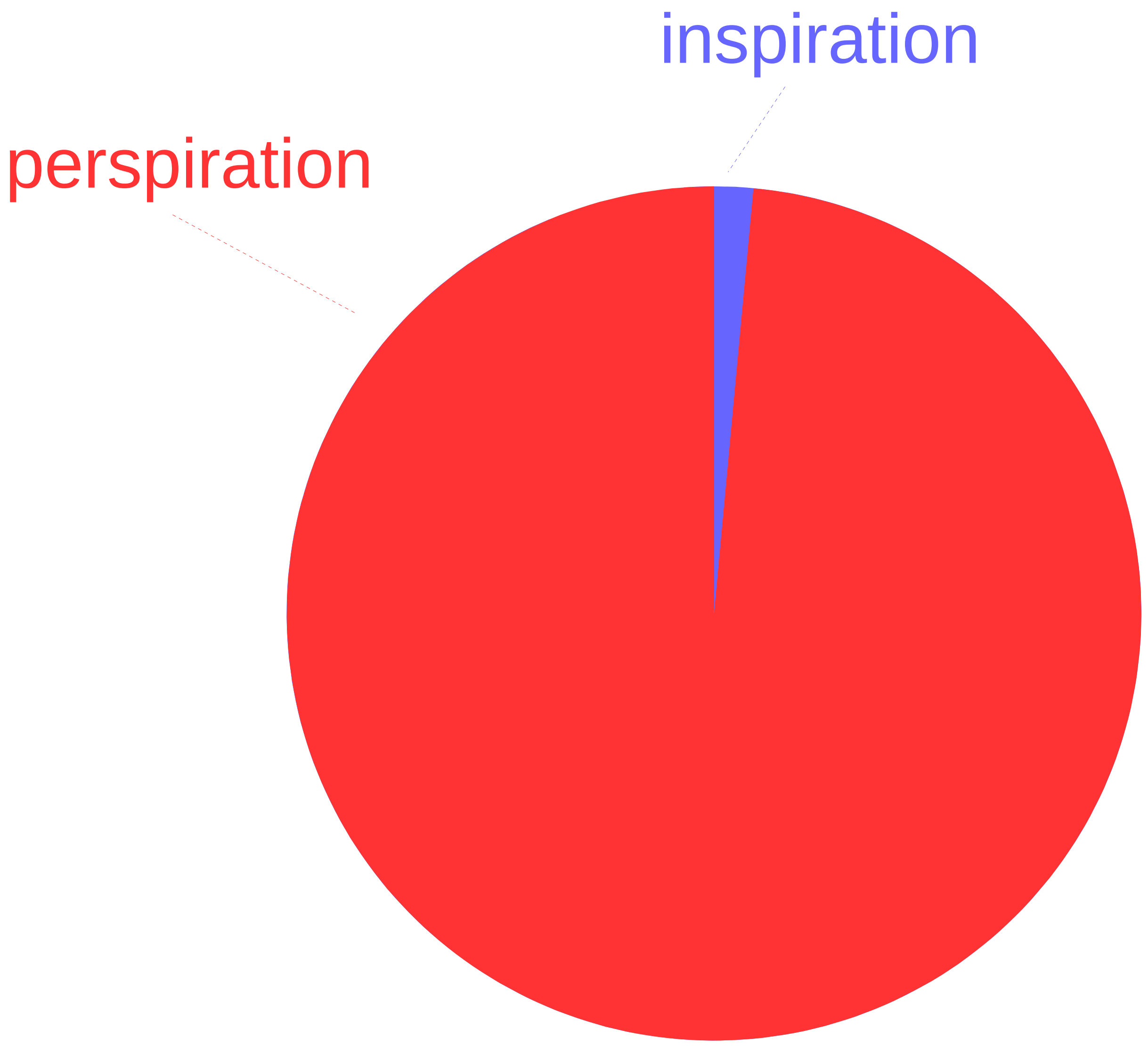
- Join the team

  … if you are up to the challenge

Summary

- Software is never done
  - … *it takes all the running you can do, to keep in the same place*
- Design the software for maintainability
  - Components, interfaces, mechanisms, testing
- Do not rely on tech trends too much
  - ♬ *That's it's all just a little bit of history repeating*
- Don’t give up, evolve!

Summary



midPoint

Conditions REST Metaroles Lifecycle Extensibility Workflow Template
Connectors Matching rules Caching Parametric roles Policy rules
Role catalog **Identity Management** Schema Expressions
Entitlements
Correlation **Synchronization** Organizational Structure
Localization Validity constraints
Scripting Self-service **Governance** RBAC LDAP Consistency
Sequences Approval Import SoD **Data Protection** LiveSync
Reporting Notifications Constants
Mappings XML/JSON/YAML Recertification Function libraries Personas
Audit Reconciliation ITSM integration Authorization Meta-data
Manual provisioning Deputy
Password management Bulk actions Dependencies Administration Web UI

For more information please visit www.evolveum.com