Evolveum®

How To Maintain <u>Million Lines</u> Of Open Source Code And Remain <u>Sane</u>*

or

The Story of MidPoint

Radovan Semančík
Rubyslava, October 2018

* More or less. More less than more.

Radovan Semančík

Software Architect at Evolveum

Architect of midPoint

Apache Foundation Committer

Contributor to several open source projects
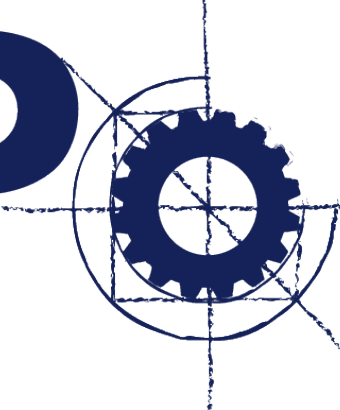
Still (more or less) sane

WARNING

HIC SUNT LEONES

Controversial statements ahead!
Political correctness (very) limited.
Mental health hazards.
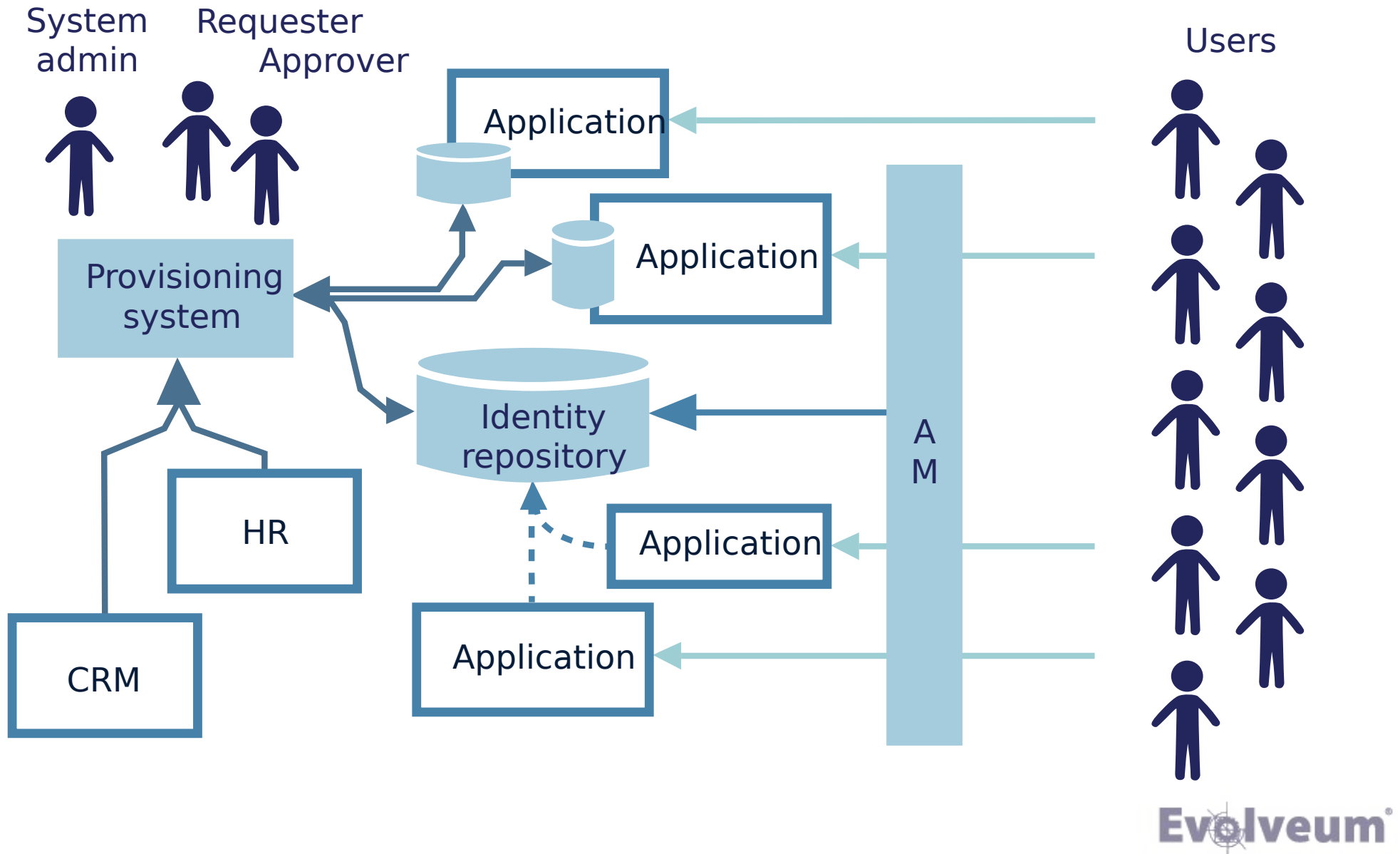Dogmatic buzzword followers should leave now.

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?

… 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 > >> ⚙

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

Dashboard

USERS	ORGANIZATIONAL UNITS	ROLES
14 enabled	19 enabled	37 enabled
16 total	19 total	38 total (+ 1 archived)

SERVICES	RESOURCES	TASKS
0 enabled	7 up	3 active
0 total	8 total	11 total

midPoint — Resource details

Resources > Resources List > Resource details — administrator

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

ADMINISTRATION
- Dashboard
- Users
- Org. structure
- Roles
- Services
- Resources
  - List resources
  - View resource
  - New resource
  - Import resource definition
  - List connector hosts
- Work items 1
- Certification
- Server tasks
- Reports

OpenLDAP — UP

Details | Defined Tasks | Accounts | Entitlements | Generics | Uncategorized | Connector

RESOURCE IS UP	MAPPINGS	SCHEMA
LdapConnector	Source and Target	3 object types
1.4.3	Synchronization defined	79 schema definitions

Capabilities

Activation | Activation Lockout | Activation Status | Activation Validity | Credentials | Password | Live sync | Test Connection | Script

Auxiliary Object Classes | Create | Update | Add/Remove Values | Delete | Read | Count Objects | Paged Search

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

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

Back | Test connection | Refresh schema | Edit configuration | Show using wizard | Edit using wizard | Edit XML

Stage: Line managers (1/3) ✓
Approver: lechuck ✓
Performer: administrator

↓

Stage: Security (2/3)
Approver: barkeeper
Approver: elaine

↓

Stage: Role approvers (all) (3/3)
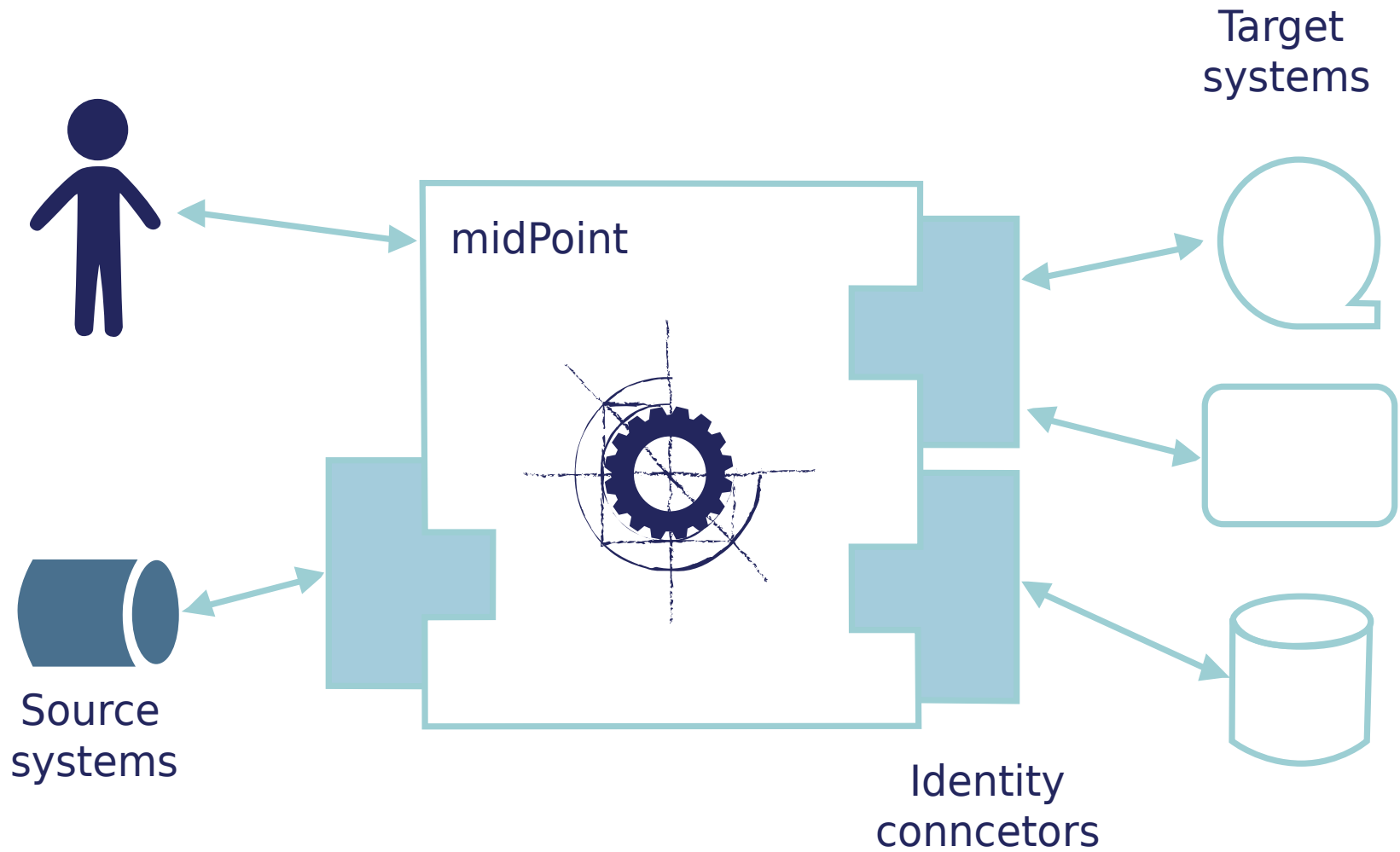Approver: cheese
&
Approver: chef
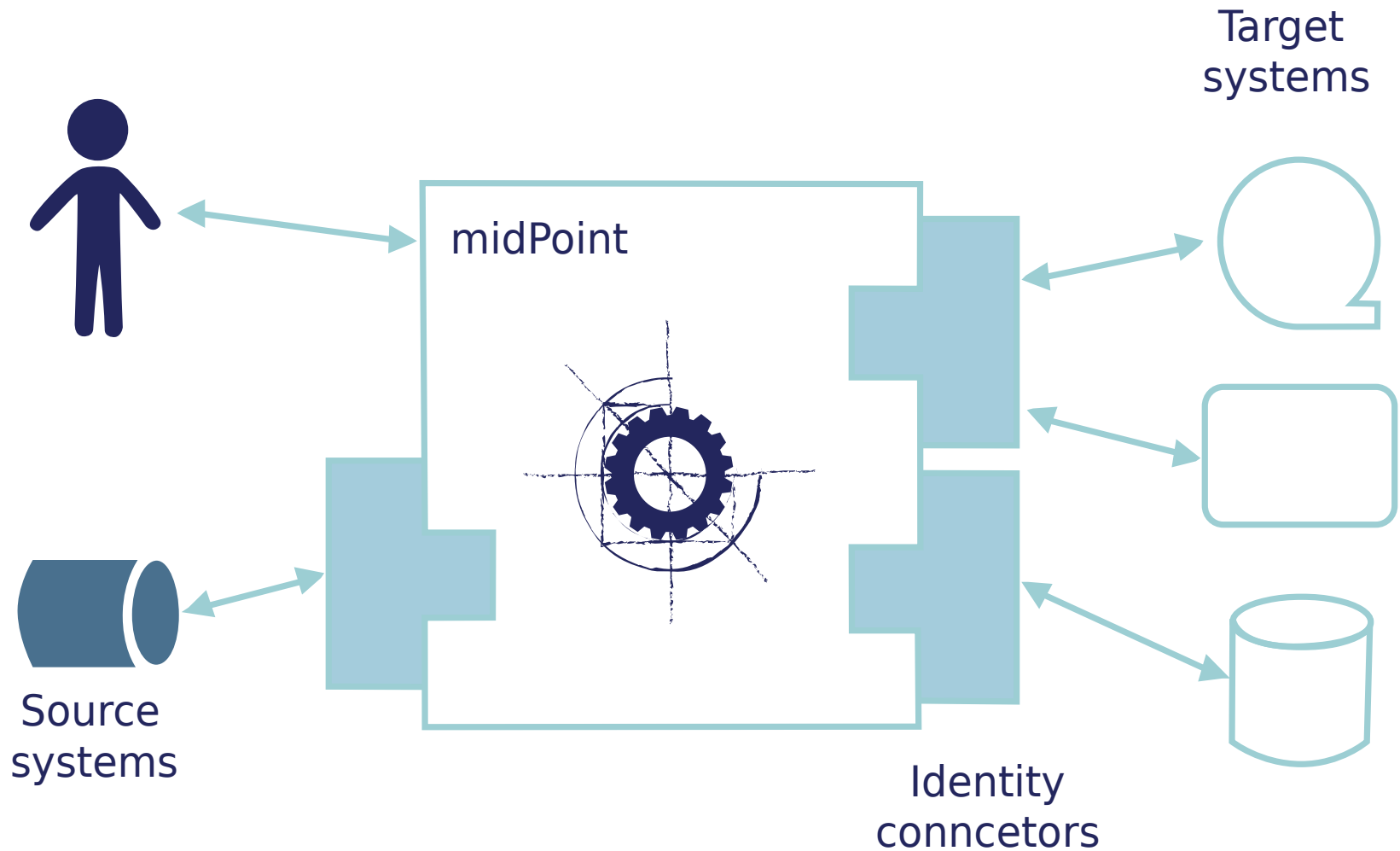
Evolveum®
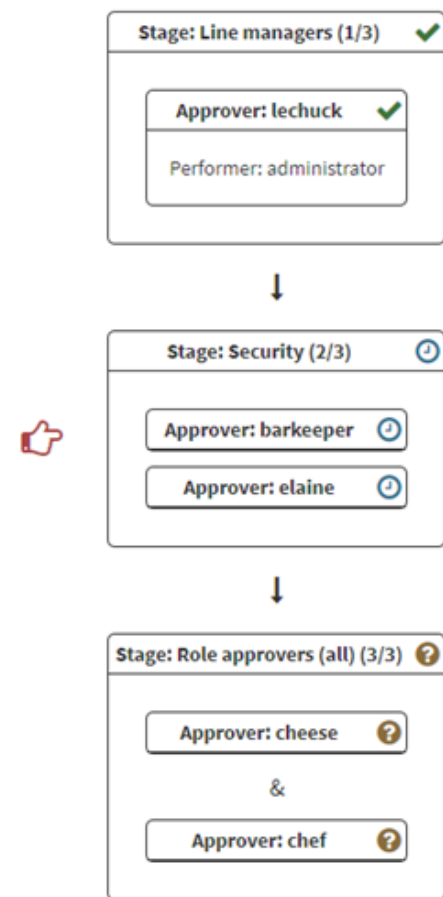
MidPoint Development

- 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)

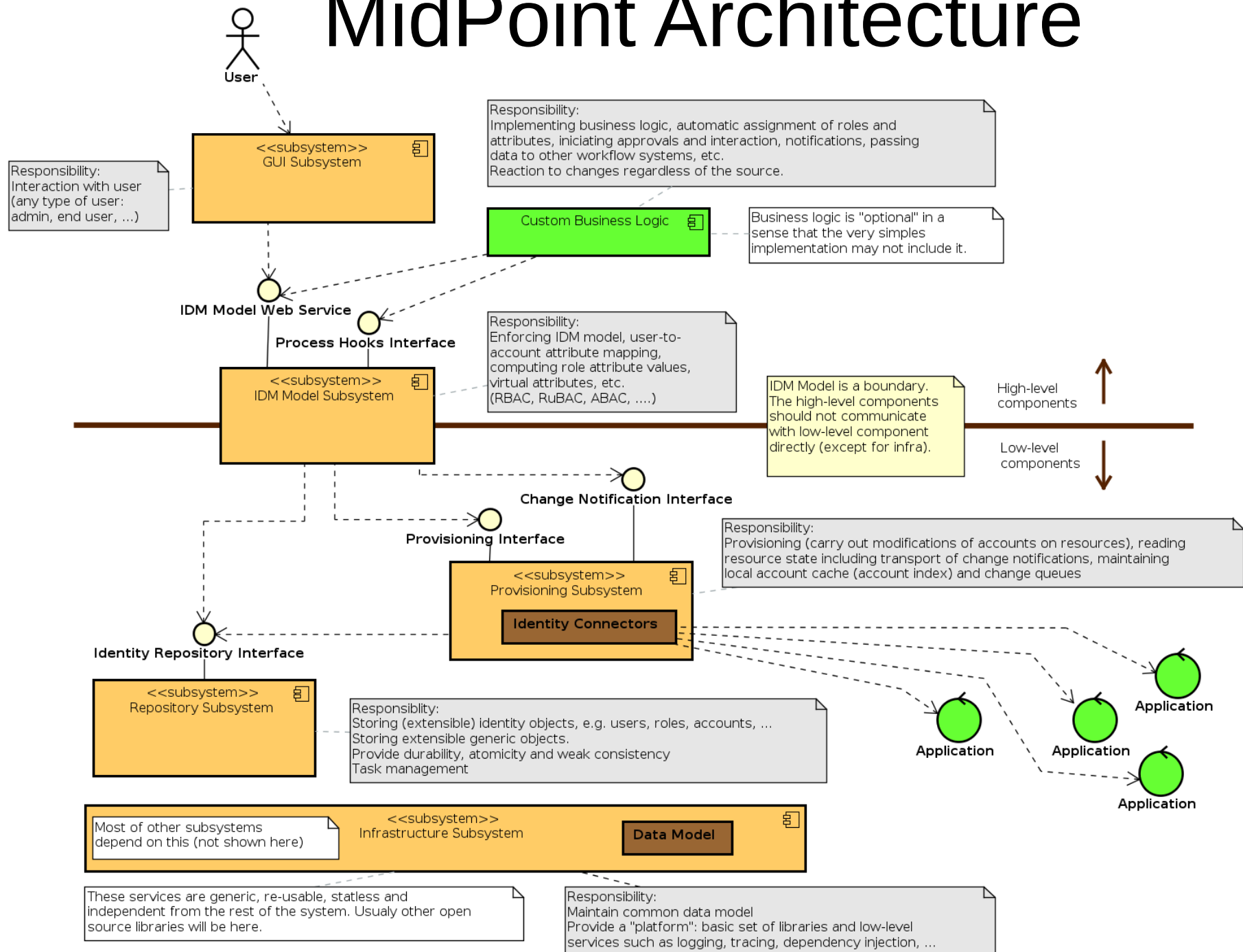- High development activity (100-200 commits/month)
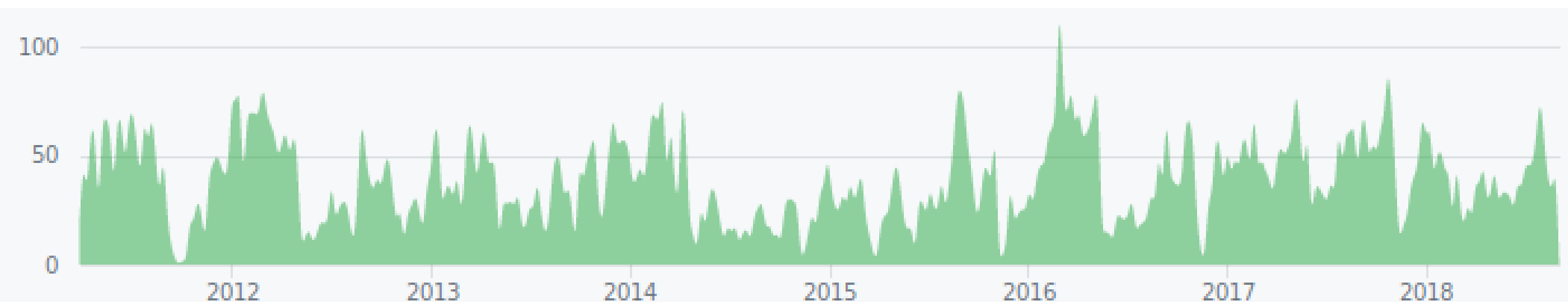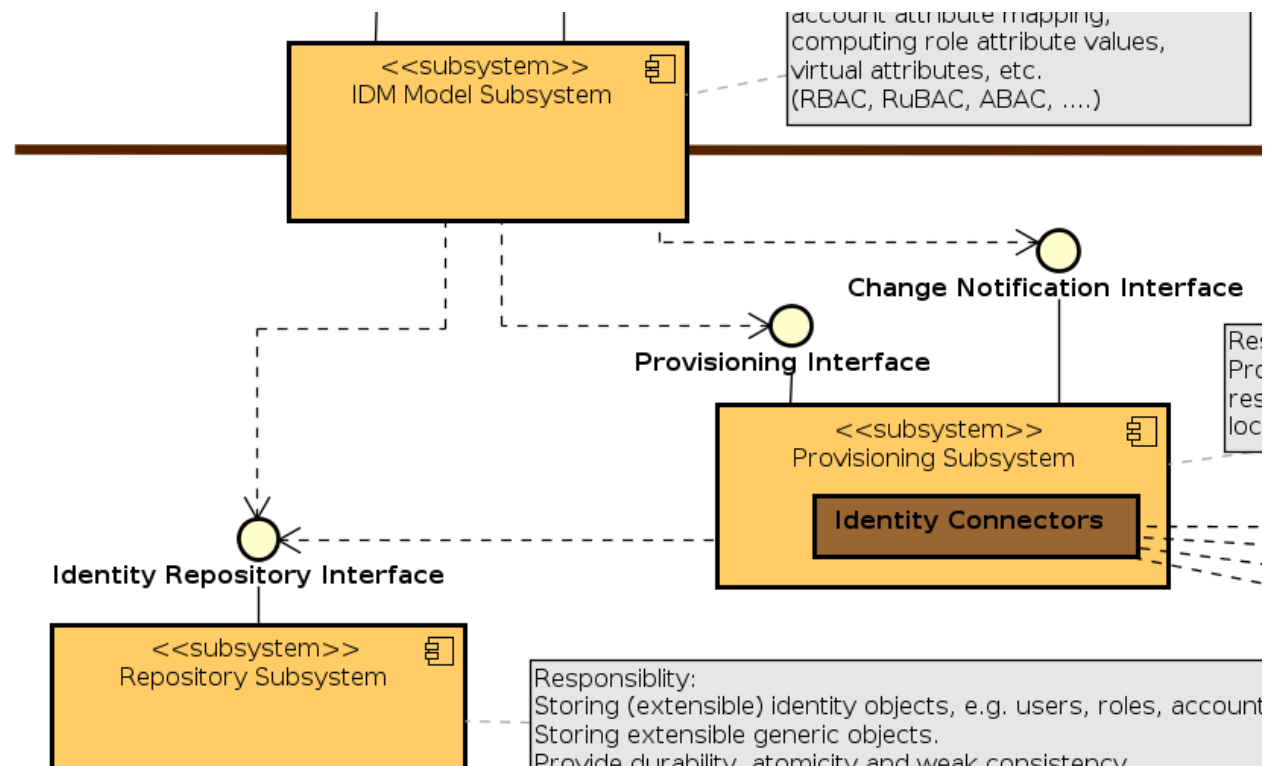
MidPoint Big Picture

midPoint

Target
systems

Source
systems

Identity
conncetors

Evolveum®

MidPoint Big Picture

Target systems

midPoint

Source systems

Identity conncetors

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

<<subsystem>>
IDM Model Subsystem

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).

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

<<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

Application

Application

Application

Application

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

<<subsystem>>
Infrastructure Subsystem

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

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)~~ : going to be replaced
- Jasper Reports : not very 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

Components, subsystems, interfaces, modules, separation of concerns

You really should pay attention in software engineering classes.
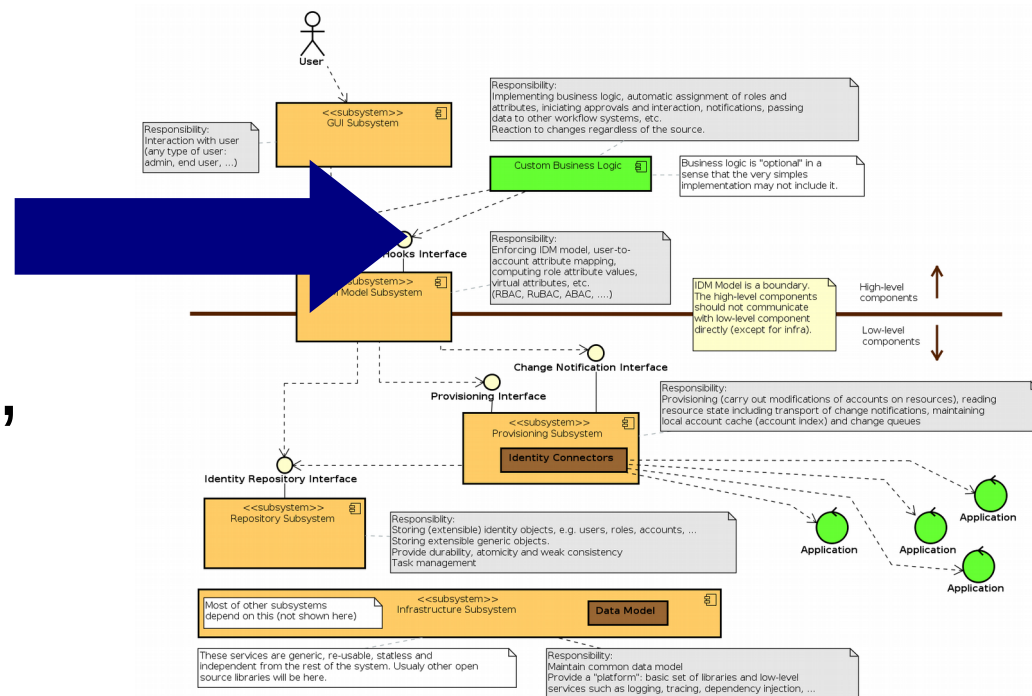
Data Model

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

Data Model
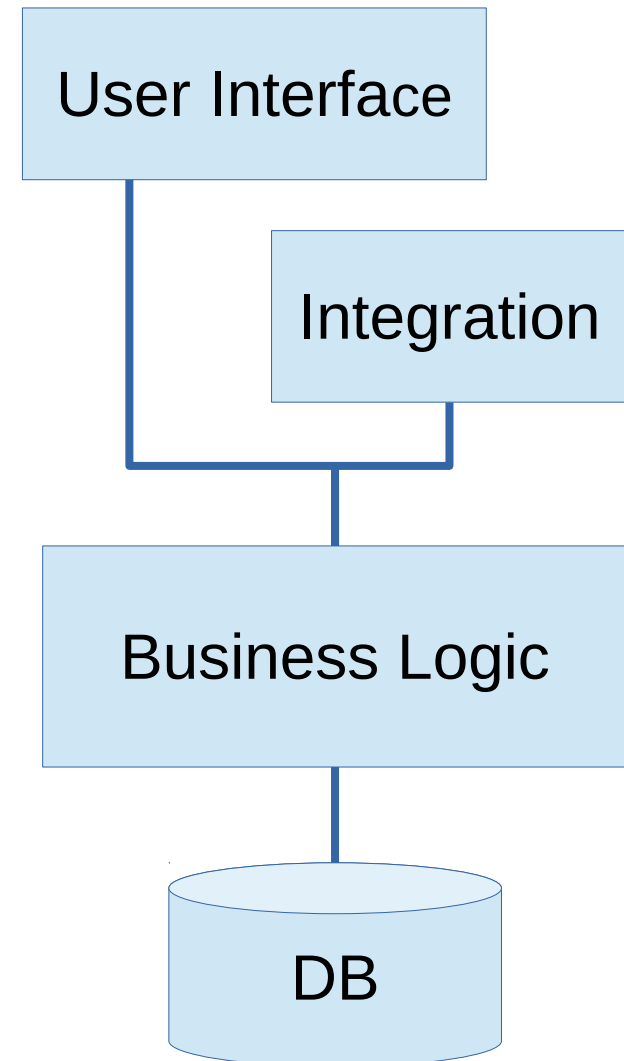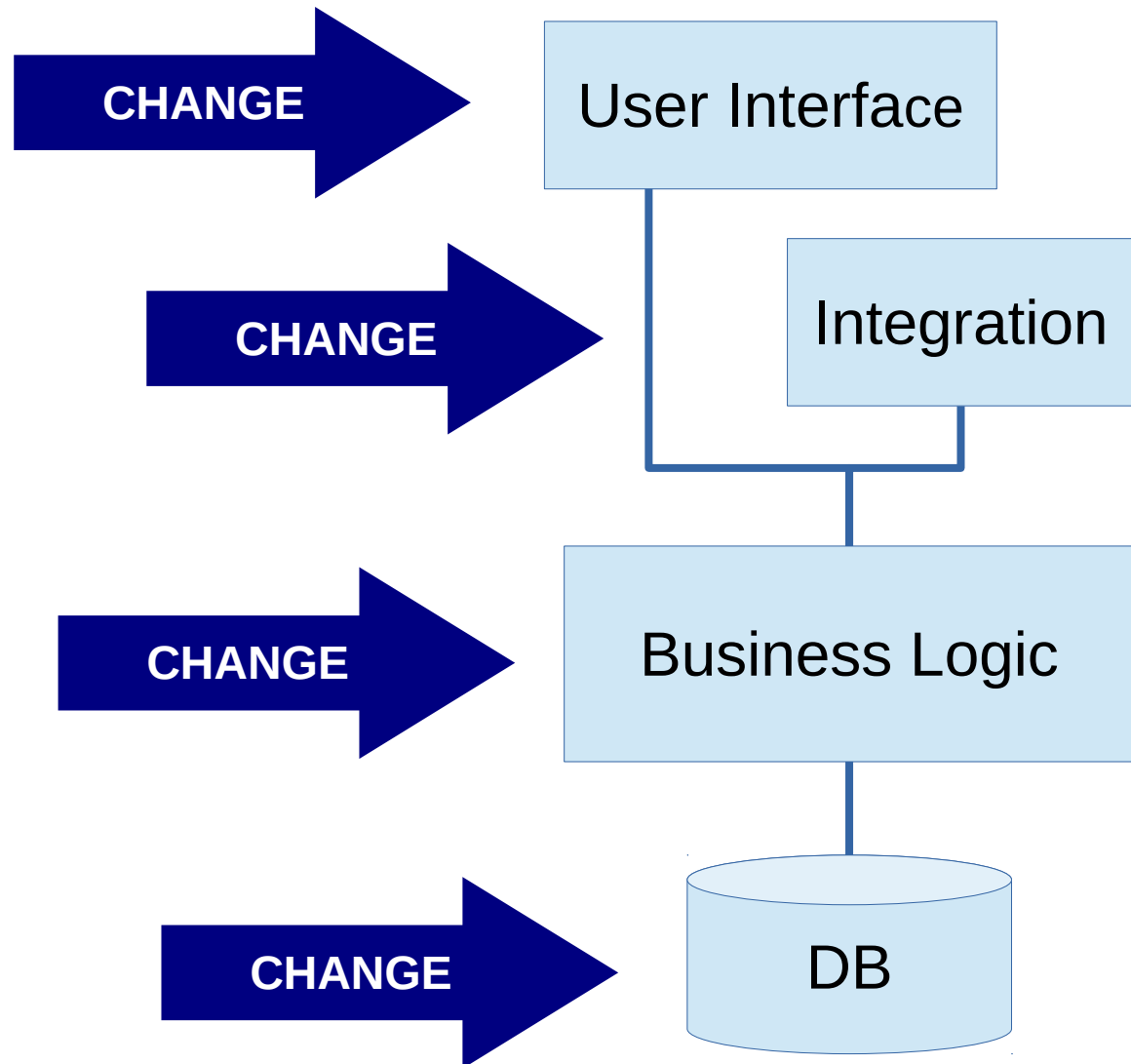
Data Model : Schema
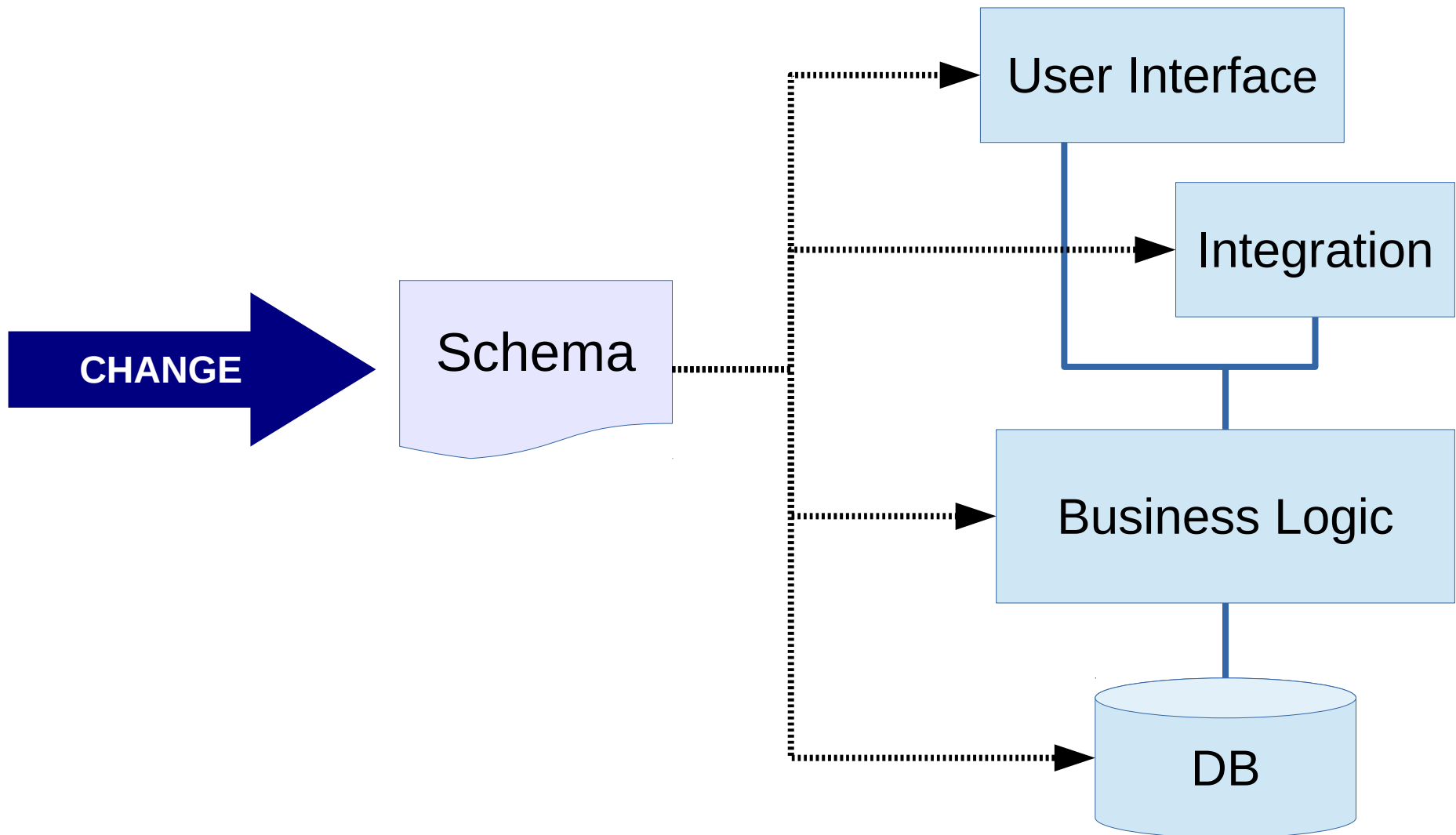
MidPoint : Prism Schema (now)
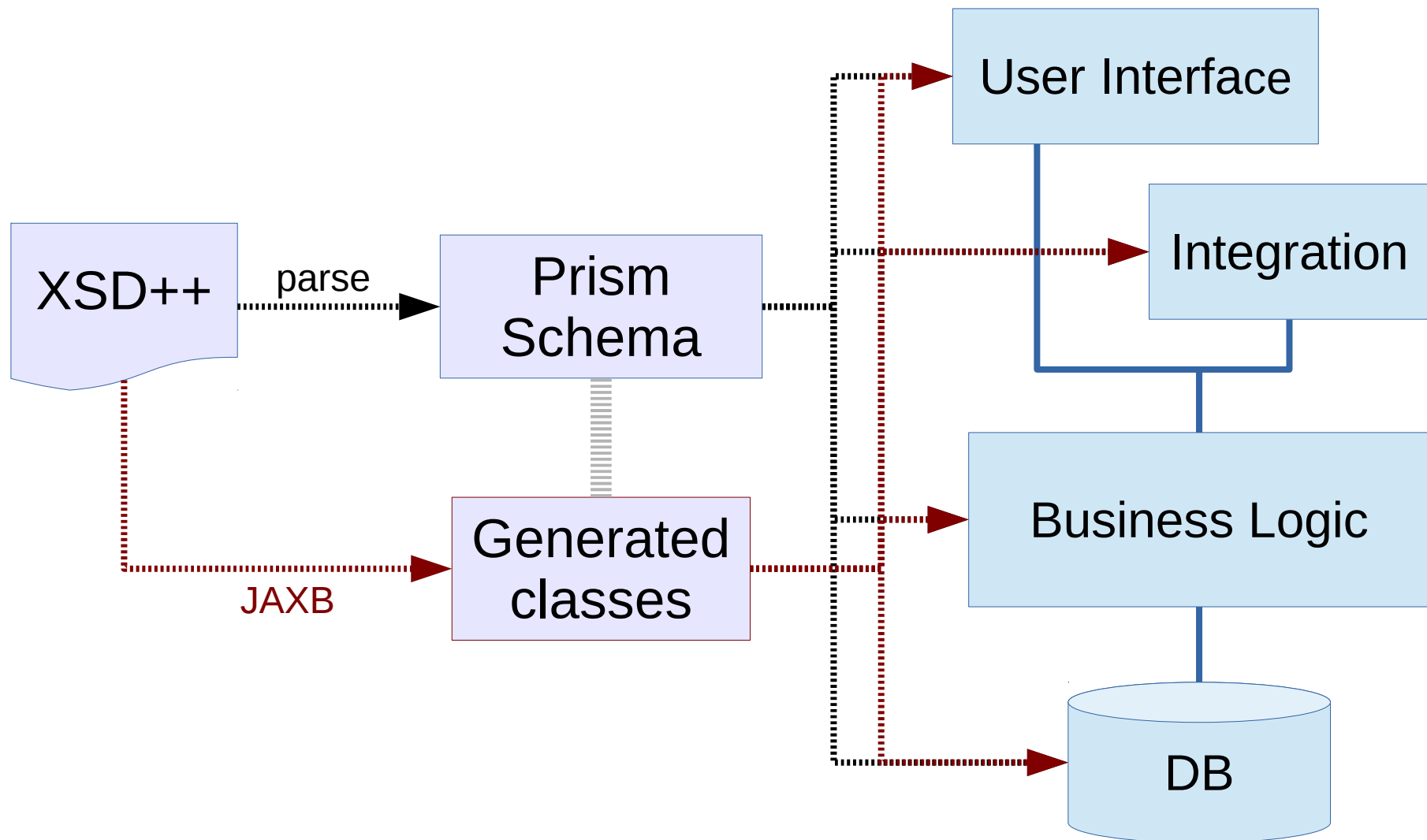
MidPoint : Prism Schema (future)

XSD++
(or whatever)

parse

Prism
Schema
+
Generated
classes

User Interface

Integration

Business Logic

DB

Evolveum®
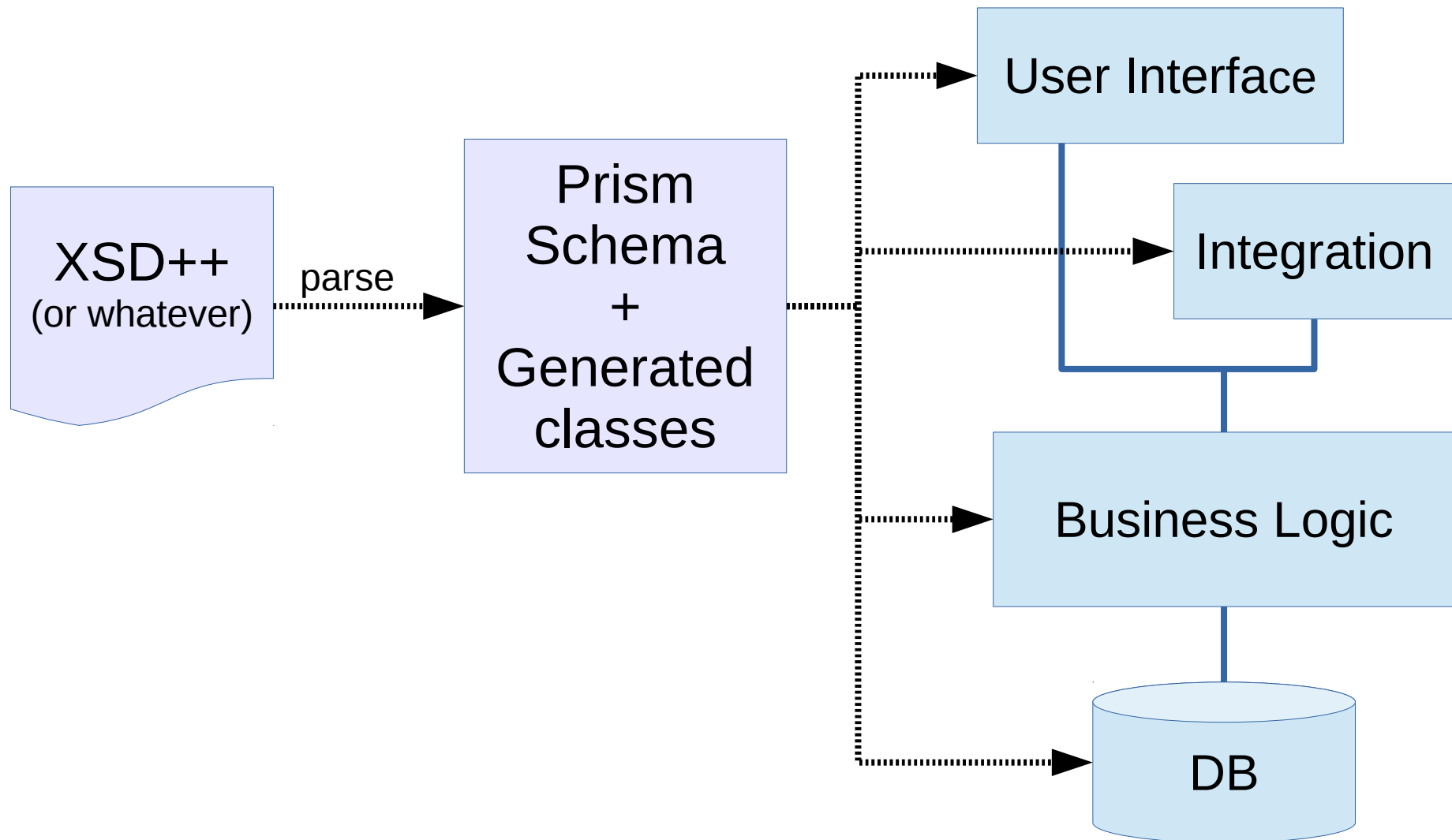
MidPoint : Prism Schema in UI

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" |

XML, JSON, YAML and Friends

Prism Object : UserType

name: foo
givenName: Foo
familyName: Bar
fullName: FooBar

Prism Schema

Parser / Serializer

```
<user>
    <name>foo</name>
    <givenName>Foo</gi
    <familyName>Bar</fa
    <fullName>Foo Bar</
</user>
```

```
{
    "name" : "foo",
    "givenName" : "Foo",
    "familyName" : "Bar",
    "fullName" : "Foo Bar"
}
```

Whatever data format
will become fashionable
next year

XML, JSON, YAML and Friends

Prism Deltas

User

oid = 8c048b2e-...
name = "jack"
fullName = "Jack Sparrow"
givenName = "Jack"
familyName = "Sparrow"
honorificPrefix = "Cpt."
employeeType = "intern"
locality = "Tortuga"

+

Object Delta

oid = 8c048b2e-...
change type = **MODIFY**
object type = User
modifications:

> **Property Delta**
> path = employeeType
> modification type = **add**
> value = **"pirate"**

=

User

oid = 8c048b2e-...
name = "jack"
fullName = "Jack Sparrow"
givenName = "Jack"
familyName = "Sparrow"
honorificPrefix = "Cpt."
employeeType = "intern"
employeeType = "pirate"
locality = "Tortuga"

User

oid = 8c048b2e-...
name = "jack"
fullName = "Jack Sparrow"
givenName = "Jack"
familyName = "Sparrow"
honorificPrefix = "Cpt."
employeeType = "intern"
employeeType = "pirate"
locality = "Tortuga"

+

Object Delta

oid = 8c048b2e-...
change type = **MODIFY**
object type = User
modifications:

> **Property Delta**
> path = employeeType
> modification type = **delete**
> value = **"pirate"**

=

User

oid = 8c048b2e-...
name = "jack"
fullName = "Jack Sparrow"
givenName = "Jack"
familyName = "Sparrow"
honorificPrefix = "Cpt."
employeeType = "intern"
locality = "Tortuga"

Big Problem of Consistency

Target systems

Source systems

midPoint

MidPoint DB

Identity conncetors

Evolveum®

Transactions / MVCC

Target systems

midPoint

YES

MidPoint DB

Source systems

Identity conncetors

Evolveum®

Transactions / MVCC

MVCC
(maybe)

midPoint

Target systems

Source systems

YES

MidPoint DB

Identity conncetors

Limited

Evolveum®

Transactions / MVCC

MVCC
(maybe)

midPoint

What is a "transaction"?

No way!

Source systems **NO**

YES

MidPoint DB

Identity conncetors

Target systems

Limited

Evolveum®
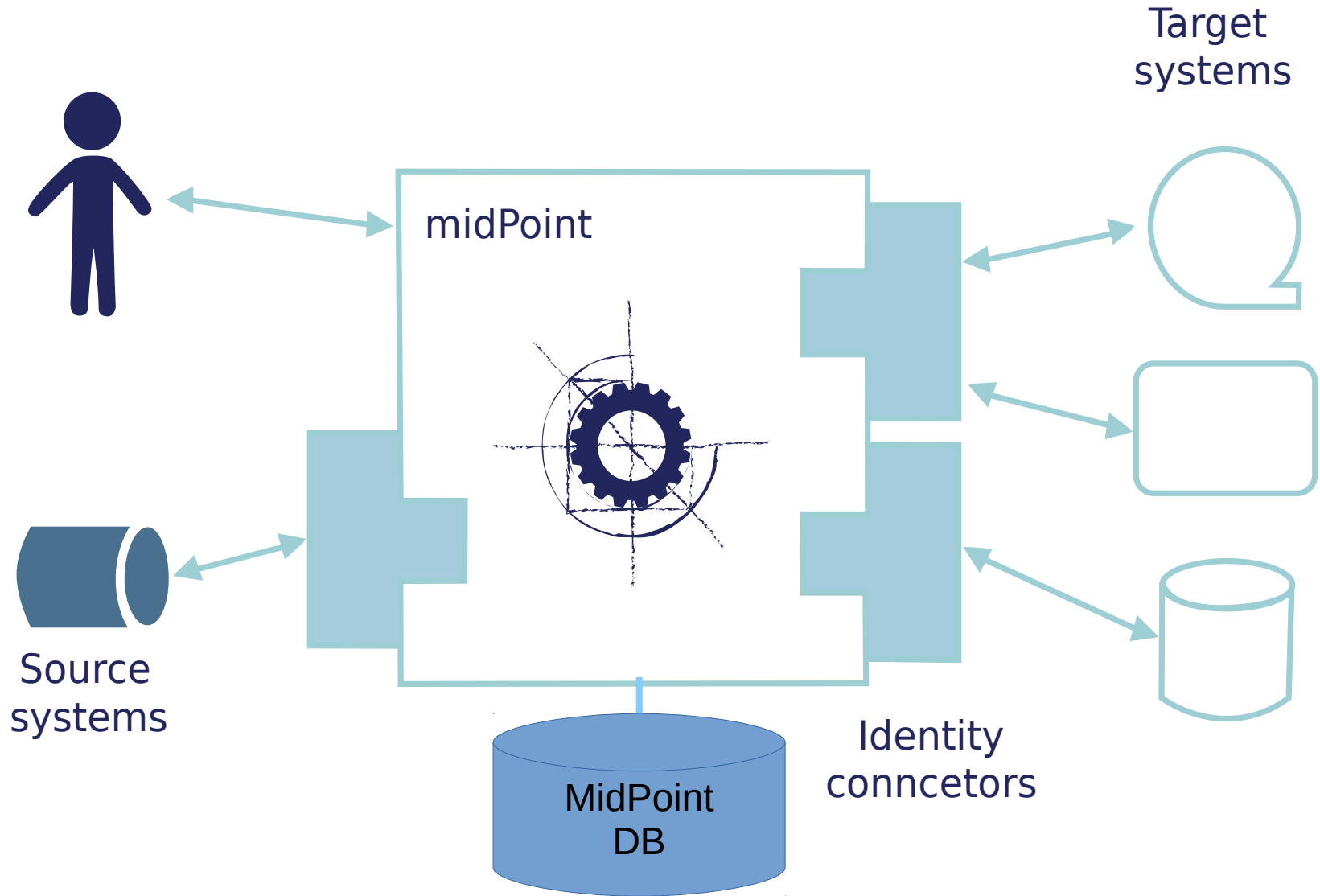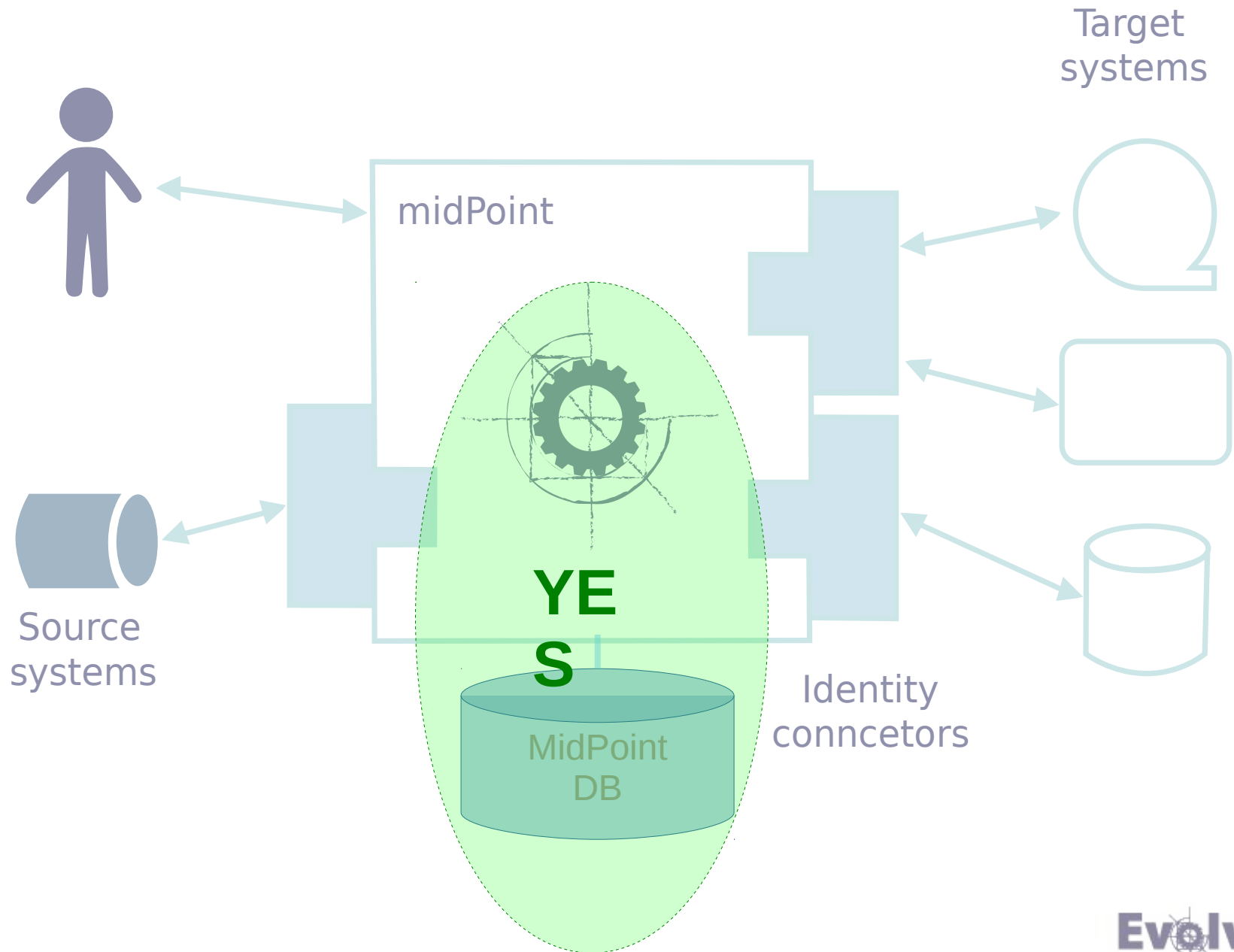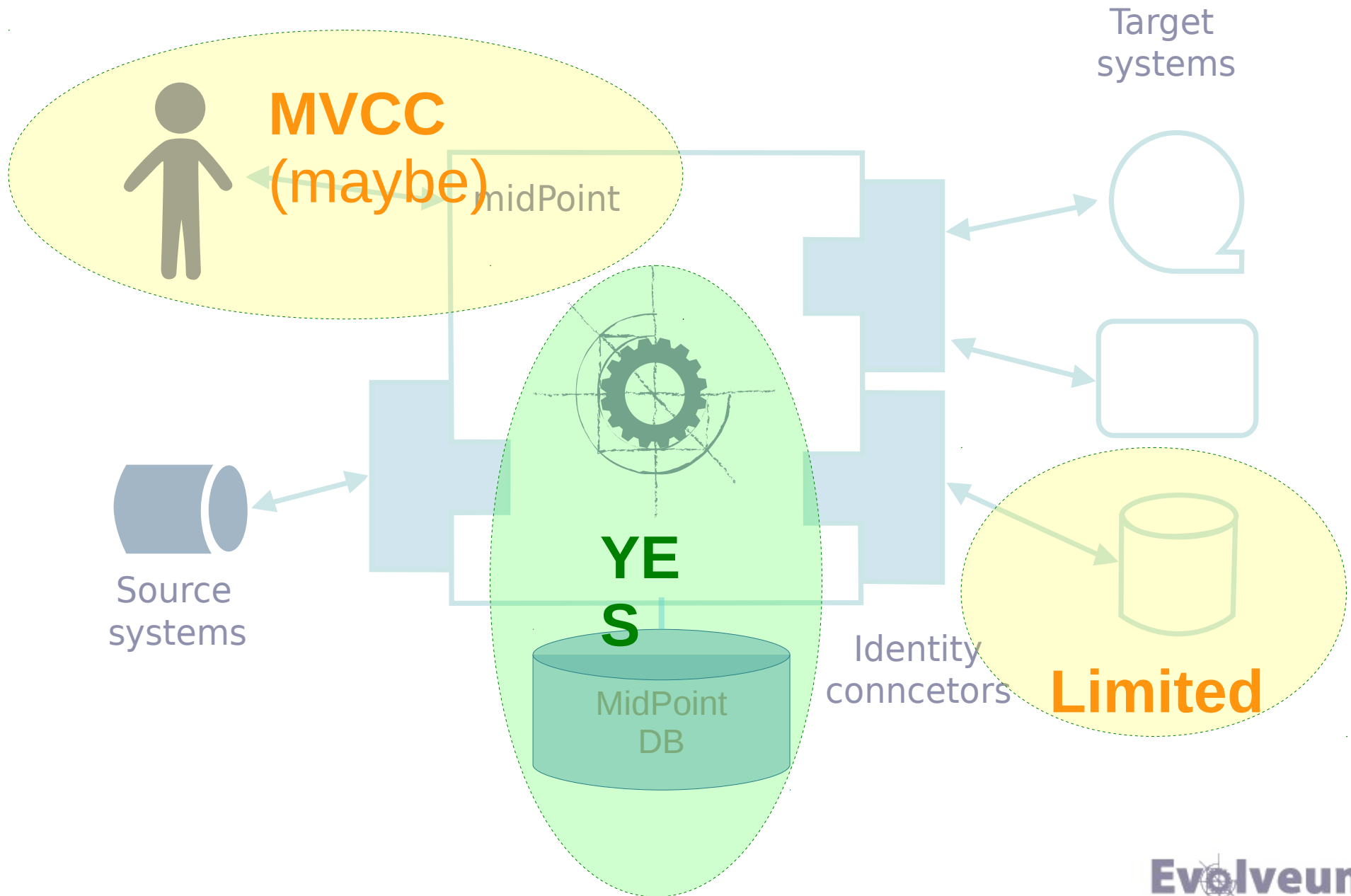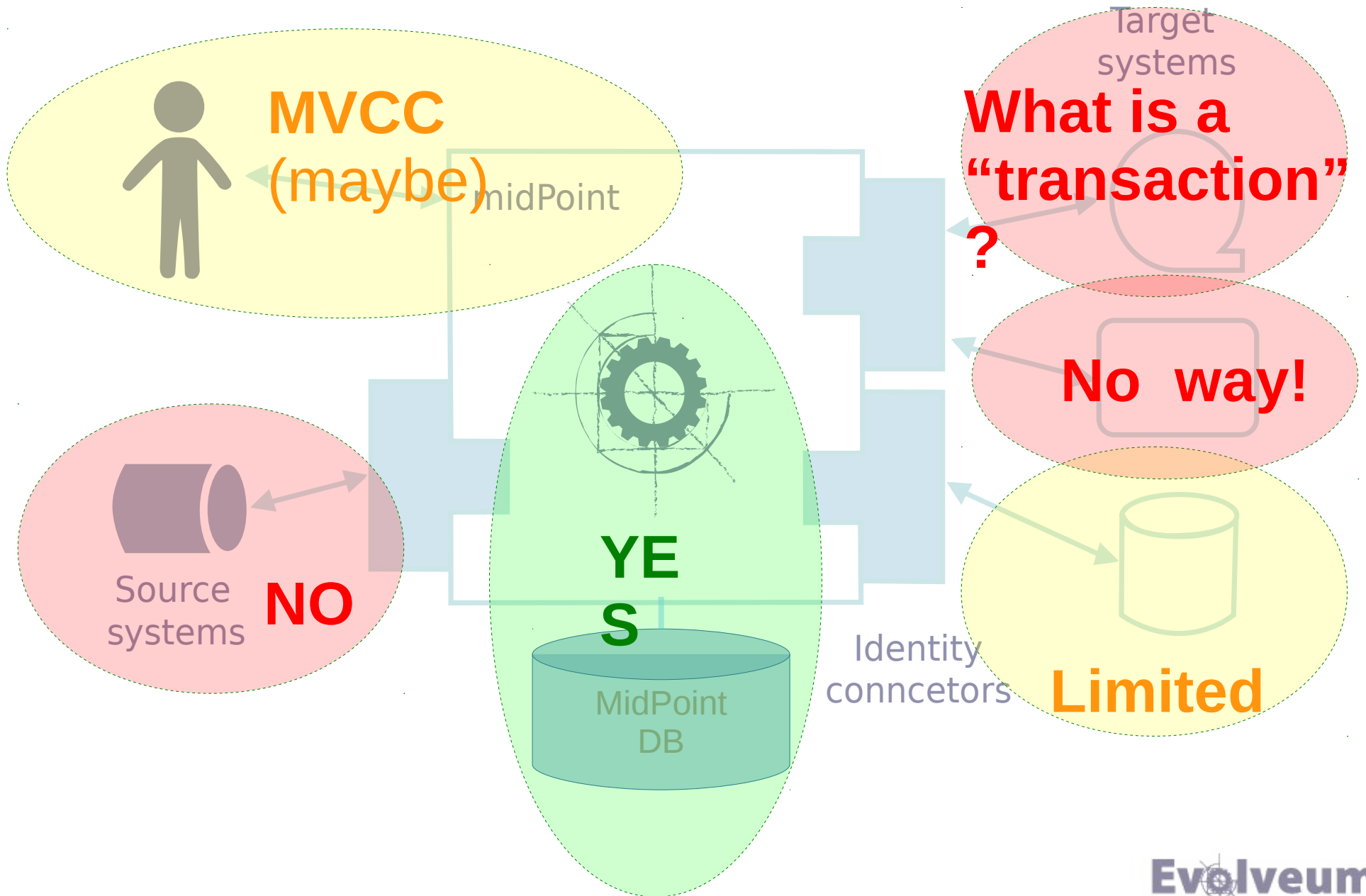
"Relativistic" Consistency

- Deltas are usually relative (*add*, *delete*)
- Apply delta in any order => equivalent value
- We need unordered multi-values for that

  … but ordering is seldom needed
- There are still some weak spots (e.g. *replace*)
- But conflicts are quite unlikely
- "Reconciliation" as safety net

Heureka! It works!

Evolveum®

Prism : Much More

- Static schema (compile-time)
- Dynamic schema (run-time)
- "Superdynamic" schema
- Raw data
    - We do not have complete schema at parse-time
- Deltas (schema-aware)
- Search filters (schema-aware, of course)
- Lifecycle (versioning, deprecated, experimental)

Questions You Surely Want To Ask

- Why XSD?
  - Because midPoint started in 2011
  - Because JSON Schema and others are equally bad
- Namespaces? QNames?
  - Yes, we use them (even in JSON and YAML)
  - No, we are not crazy (yet)
  - End user (usually) does not need to deal with them
  - QName == URI
  - Benefits: extensibility, versioning

Evolveum®

RESTful API

`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

Evolveum®

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 … 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

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.

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

<<subsystem>>
GUI Subsystem

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

<<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

Application

Application

Application

Application

Application

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

<<subsystem>>
Infrastructure Subsystem

Data Model

These services are generic, re-usable, statless and

Responsibility:

Rolling-Wave Approach

2018	2019		2020		2021
v3.9 exact plan	v4.0 rough plan	v4.1 some plan	v4.2 maybe	v4.3 probably	??? v5.0 here or maybe not

2018	2019		2020		2021
v3.9 done	v4.0 exact plan	v4.1 rough plan	v4.2 some plan	v4.3 most likely	v5.0 here maybe

2018	2019		2020		2021	
v3.9 done	v4.0 done	v4.1 exact plan	v4.2 rough plan	v4.3 some plan	v4.4 maybe	v5.0 probably

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

Evolveum®

Questions you wanted to ask at the beginning

- 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 (remember: code generated from data model)
  - Old language +1: libraries for everything
  - Old language -1: you need to avoid landmines
  - OpenJDK
  - Hindsight: Java is lesser evil

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
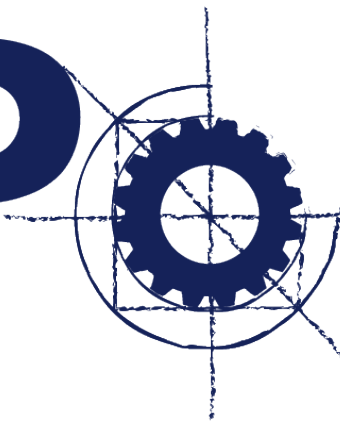 - Professional services, projects (minimal) → partners

Summary

midP⊙int
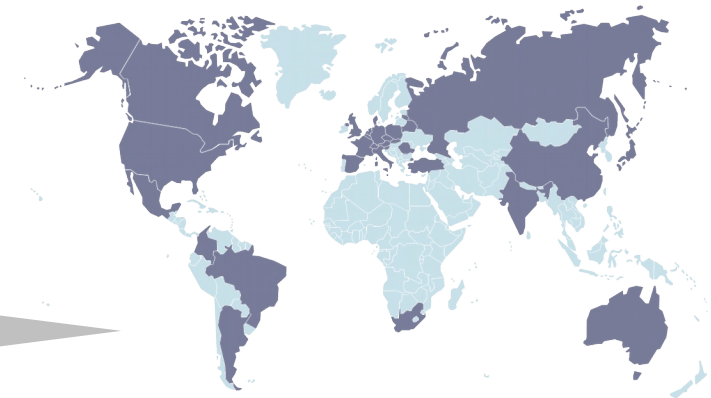
- Million lines of Java code, 7 years, small team

  … and still going fast and strong.

- Open source, self-funded … and survived!

- Good architecture, rolling-wave design

- Schema-aware from bottom to top

- Not entirely normal project

Evolveum®

Join the Team

- Java developers, IDM engineers, … marketing

- Bratislava, Košice

  ... or anywhere (remote work)

- Join the team

  … if you are up to the challenge

midPoint

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

For more information please visit
www.evolveum.com