Software Architecture

Theory and Practice

Evolveum

Radovan Semančík

April 2018

Who Am I?

Ing. Radovan Semančík, PhD.

Software Architect at Evolveum

Architect of Evolveum midPoint

Apache Foundation committer

Contributor to ConnId and Apache Directory API

Poll

Who wants to be:

1. Coder/developer
2. Software designer/architect
3. Manager

So, you wanna be an architect?

What Does Software Architect Do?

Theory

- Draw diagrams (UML anyone?)
- Design great and important systems
- Be a big boss

What Does Software Architect Do?
Practice

- ~~Draw diagrams (UML anyone?)~~

  … implement it too. And test. And document.

- ~~Design great and important systems~~

  … more like databases and JavaScript.

- ~~Be a big boss~~

  … in fact do many things by yourself.

architecture

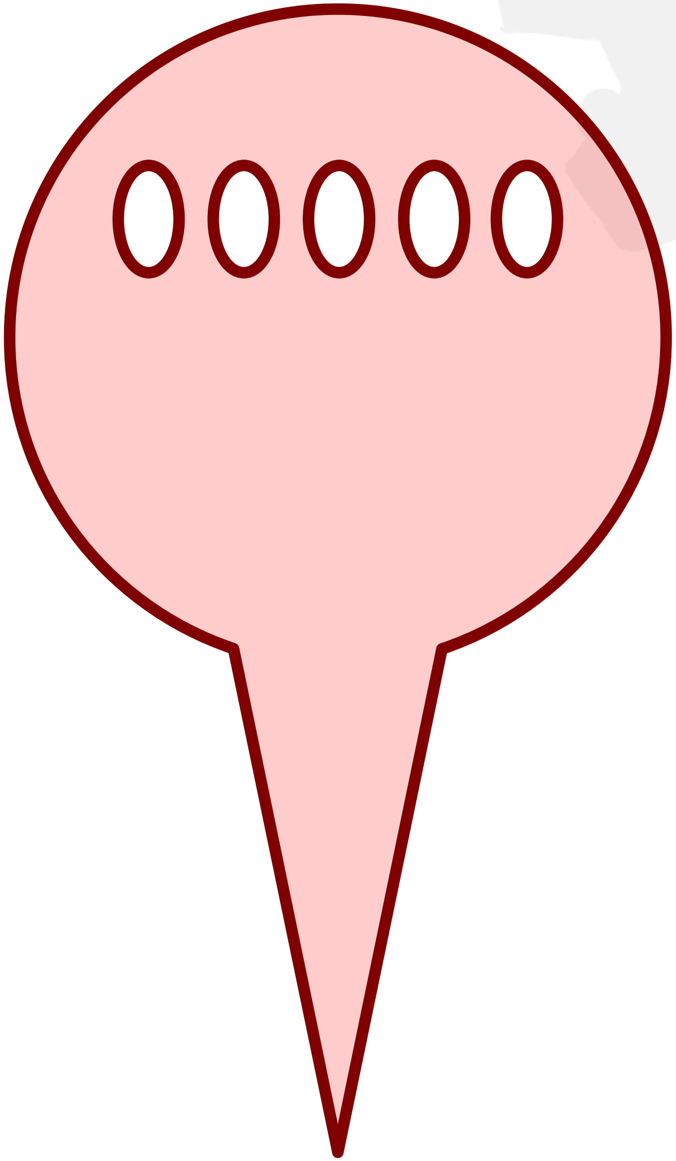
The art or science of building; especially, the art of building houses, churches, bridges, and other structures, for the purposes of civil life; -- often called civil architecture.

Construction, in a more general sense; frame or structure; workmanship.

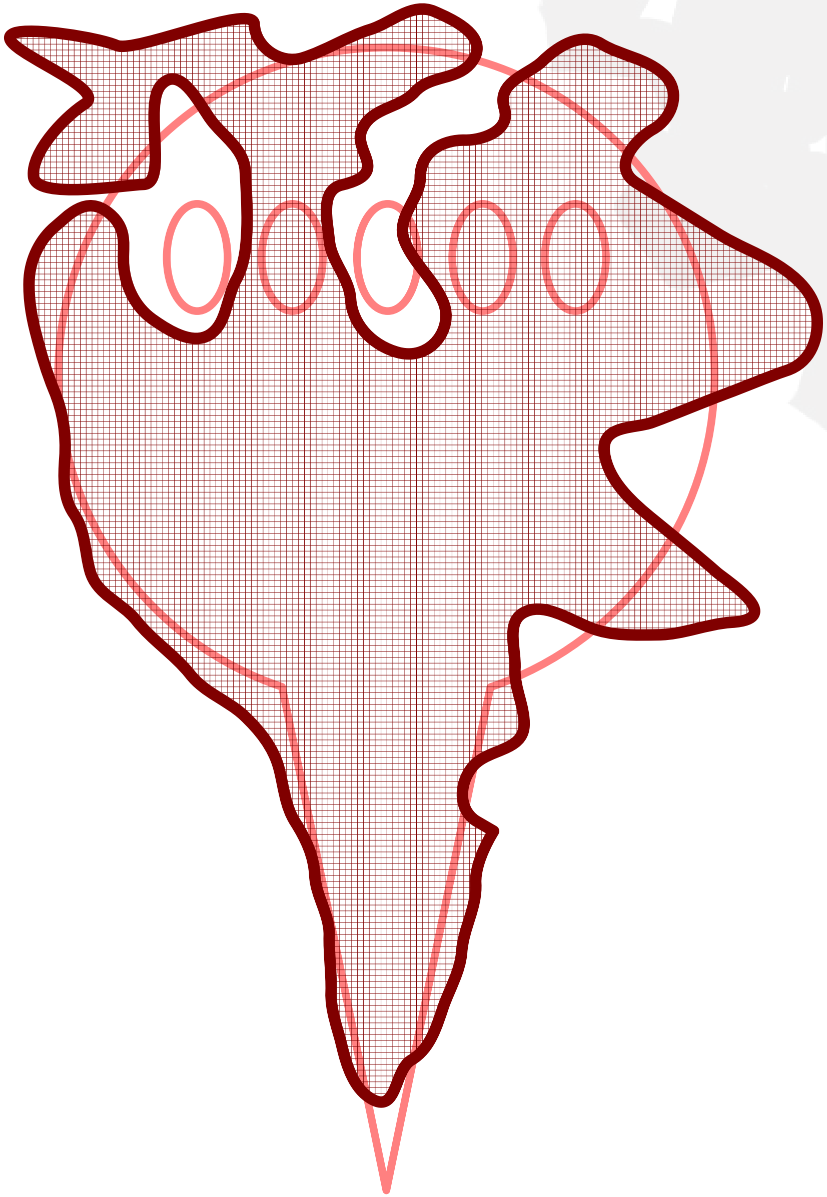
Webster, 1913

How it all works in practice ...

What Client Wanted

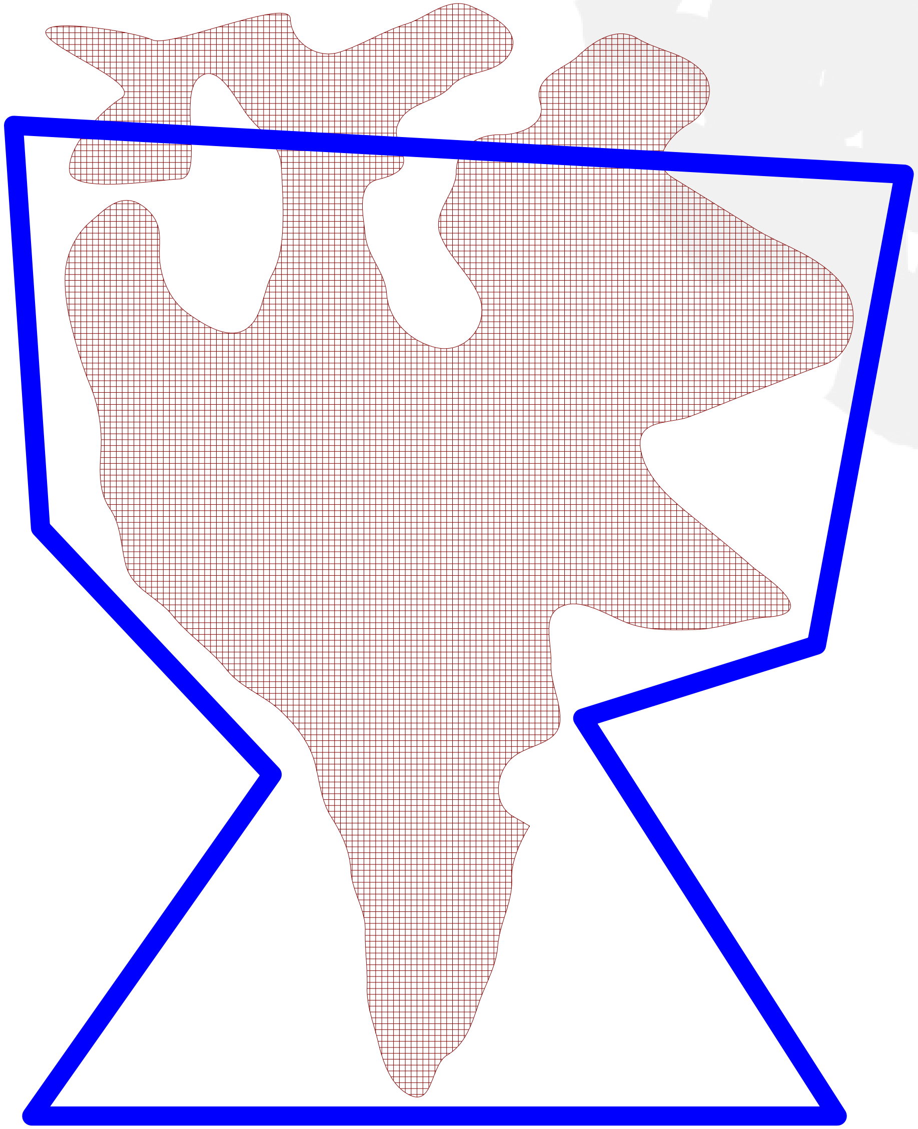


What Client Described

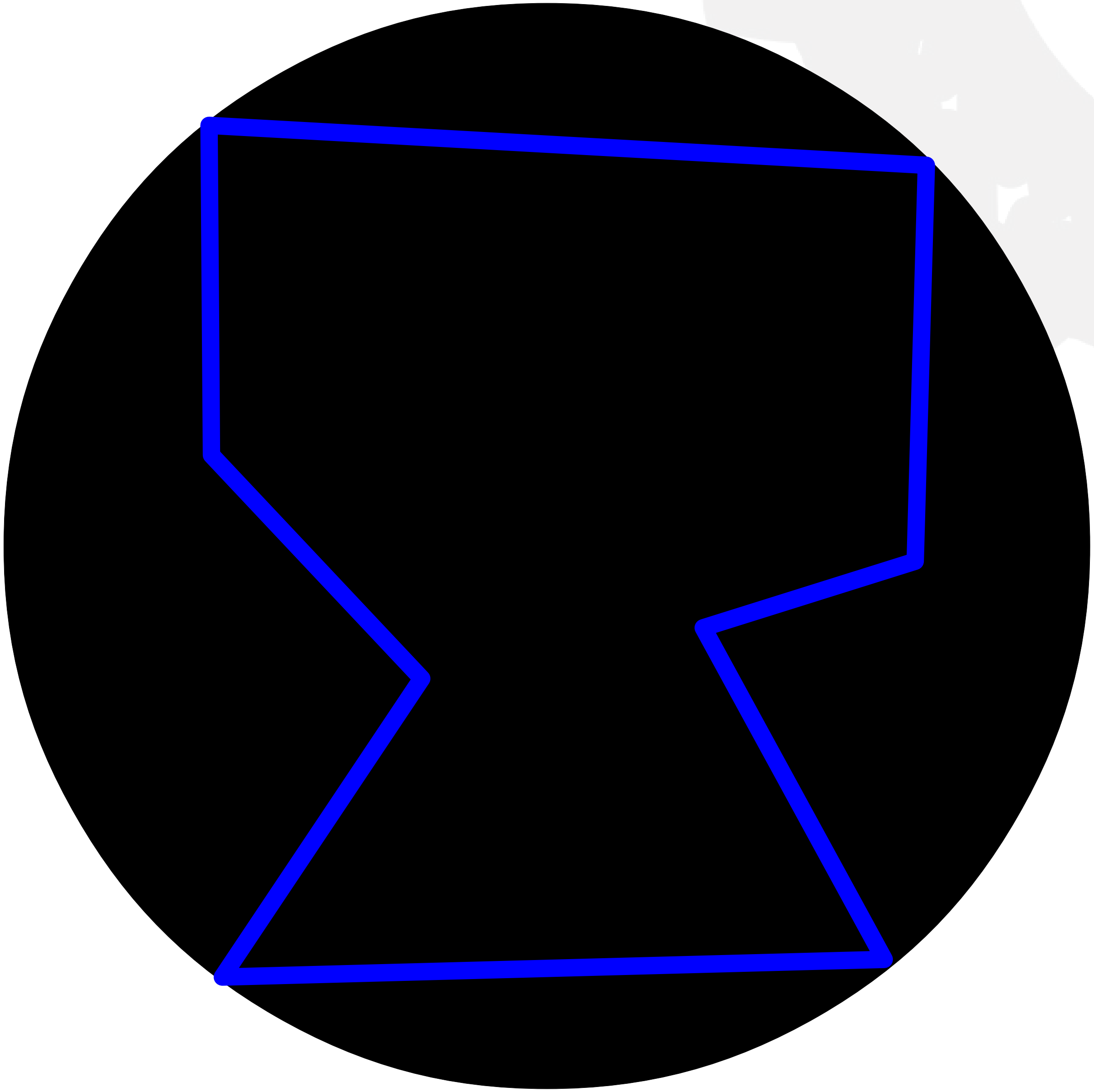


Evolveum

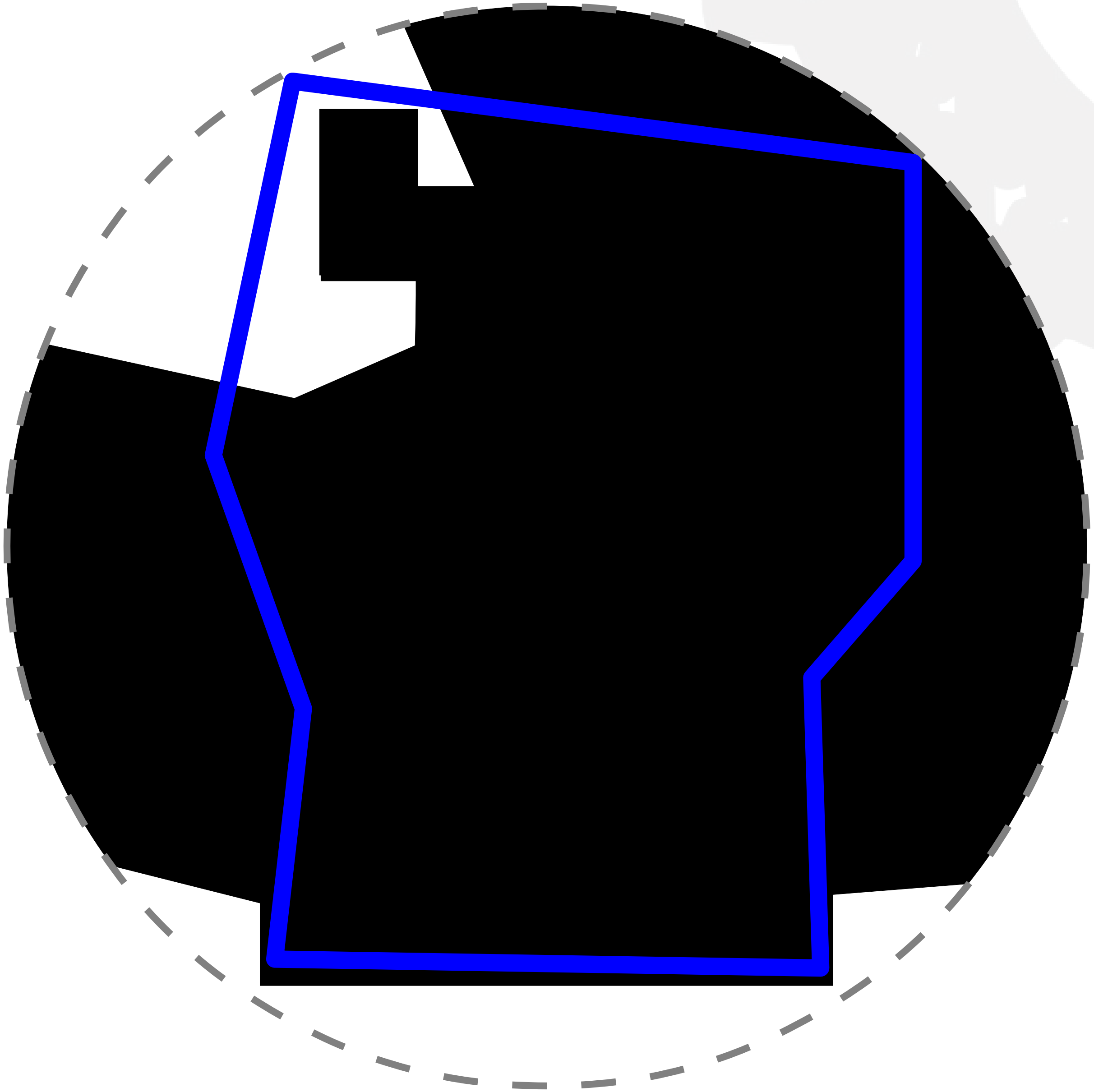
How Architect Understood



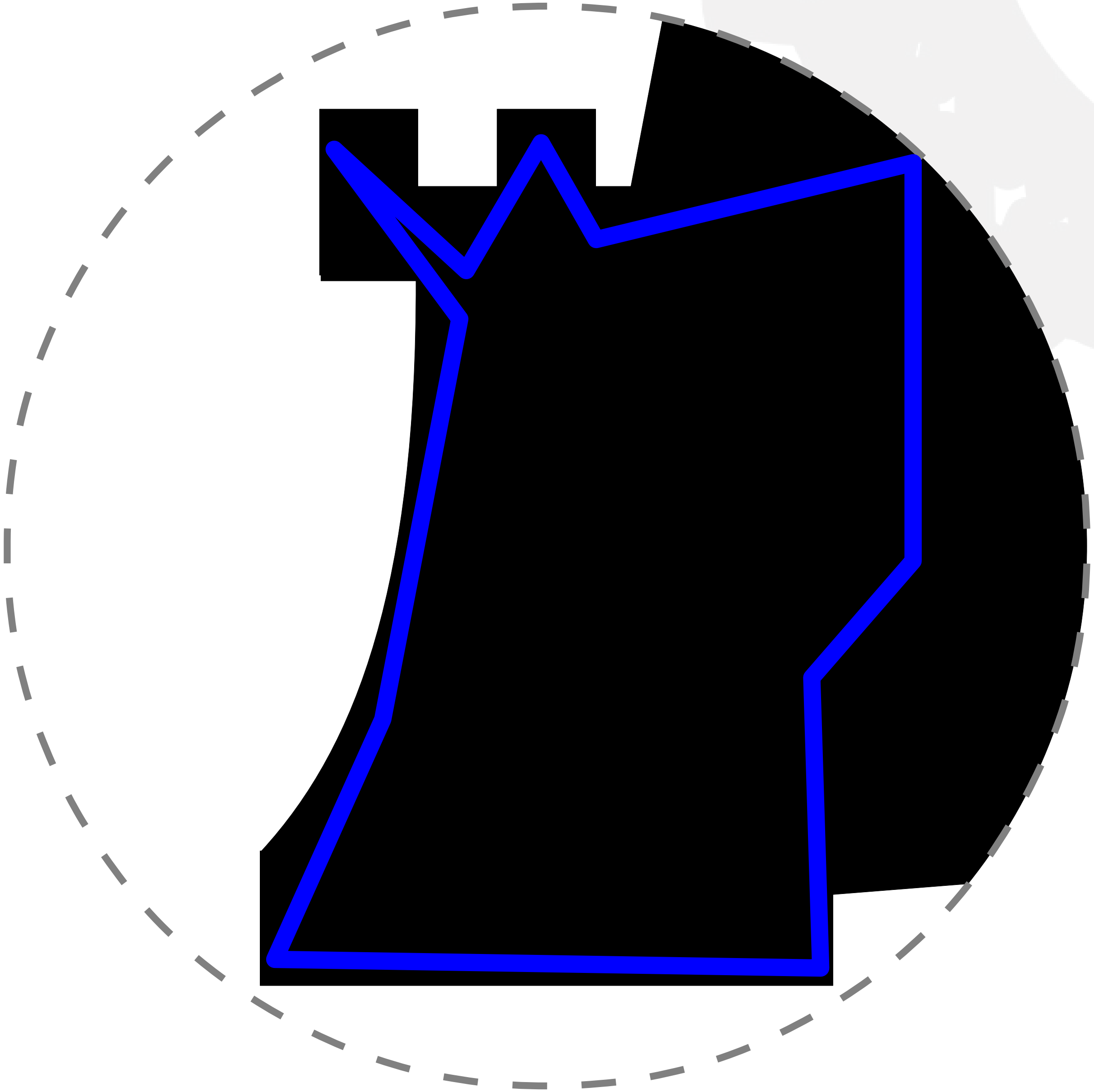
Empty Set of Constraints



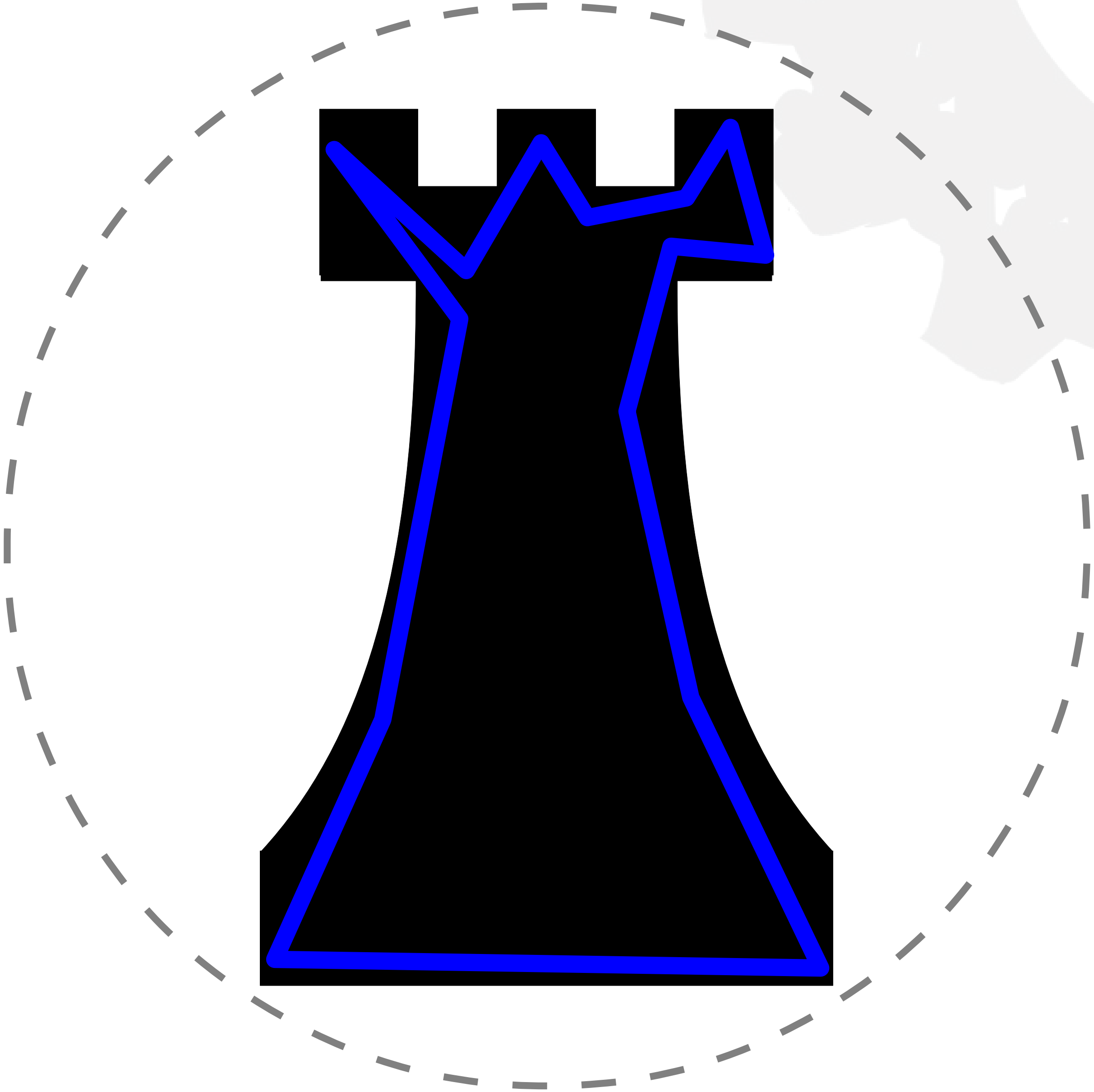
Adding Constraints



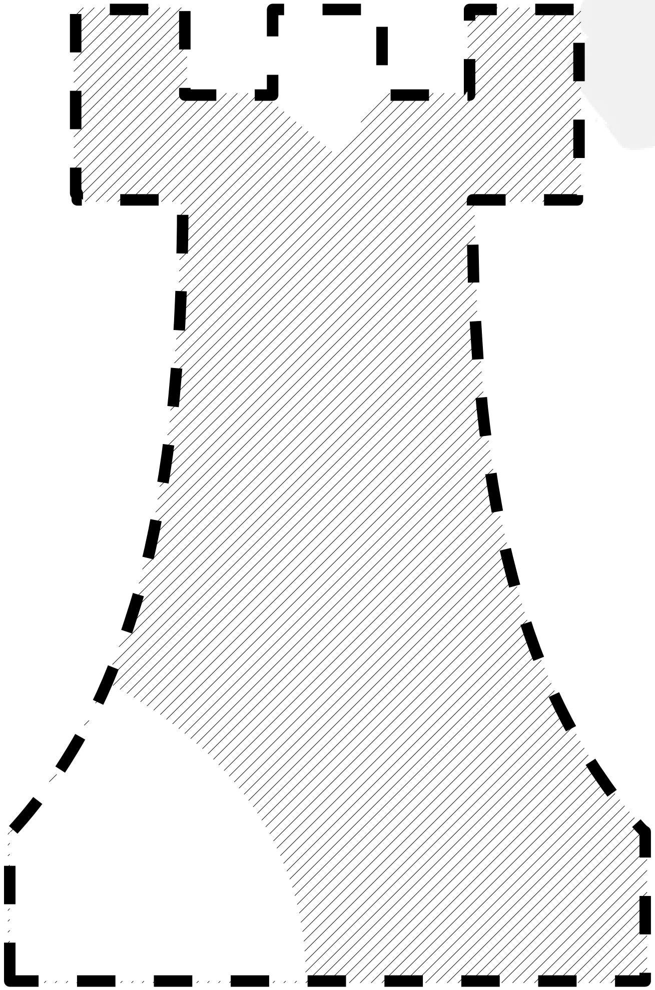
Adding Constraints



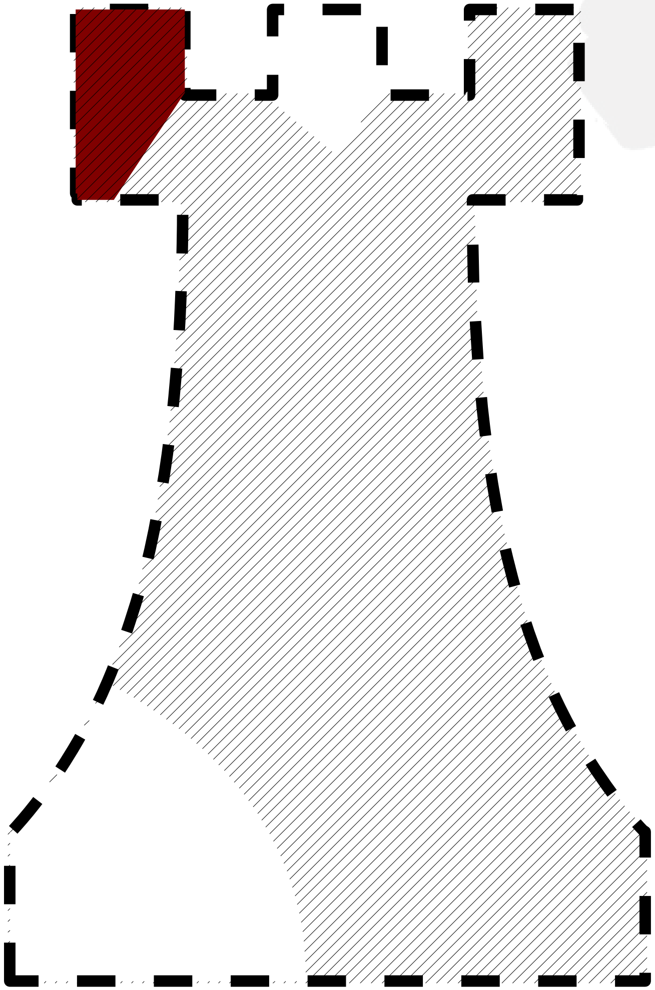
Architecture Finished



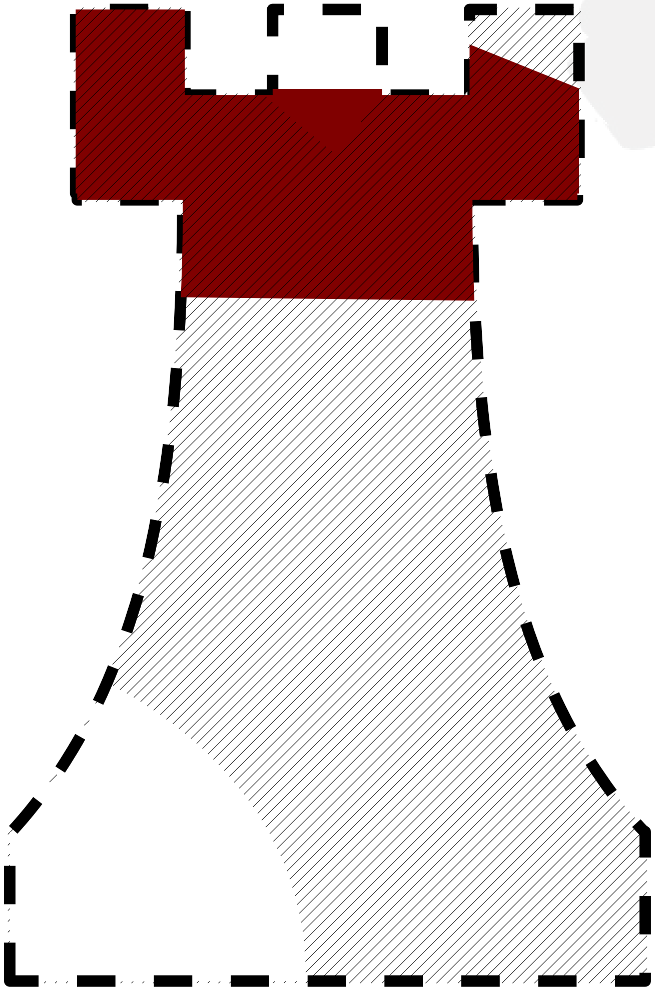
Architecture Documented



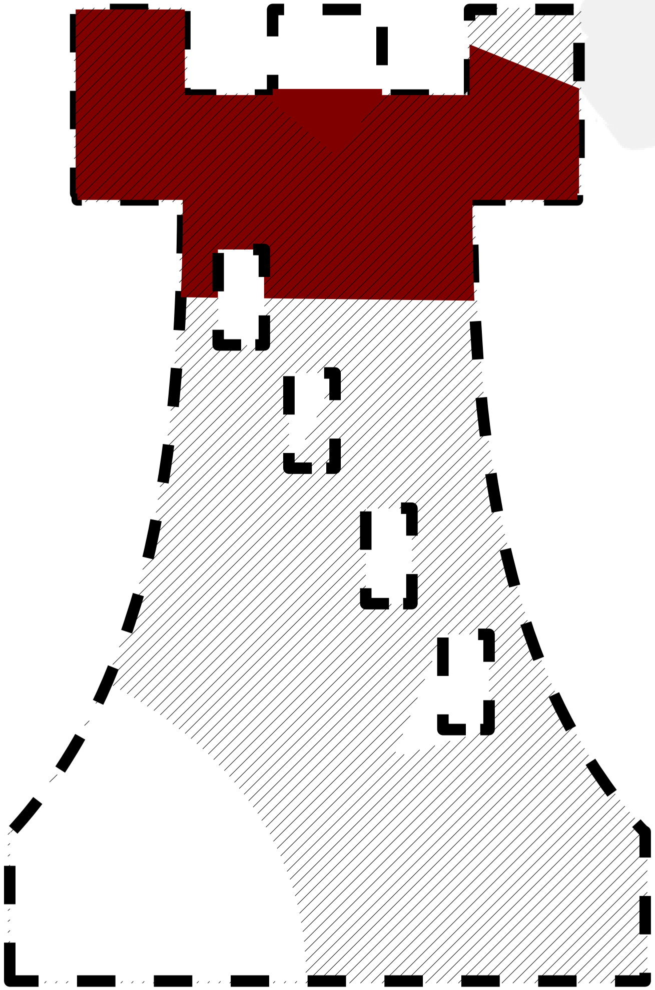
Start of Development



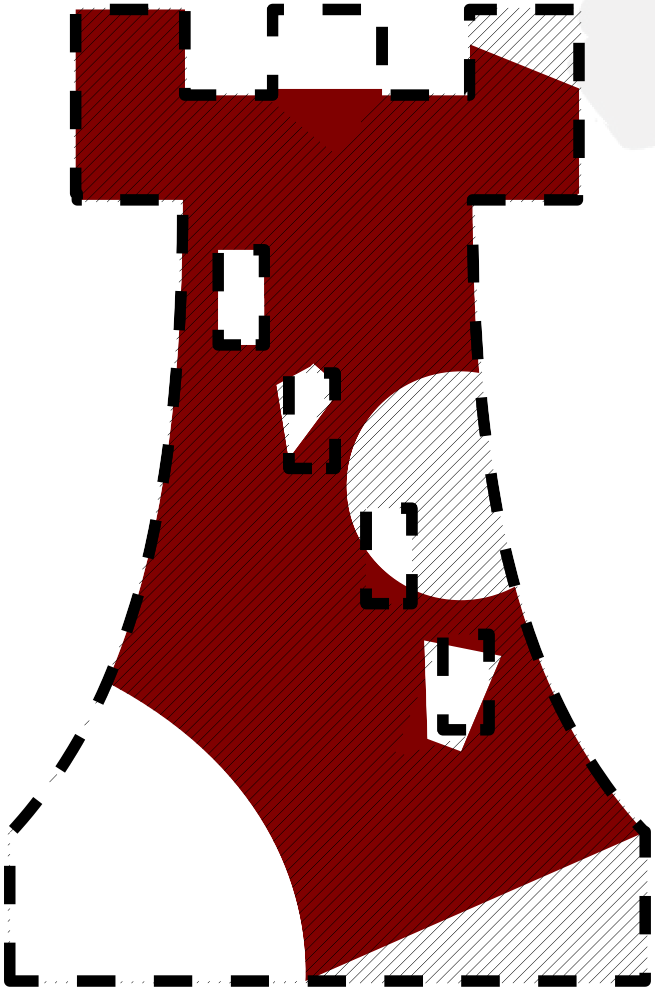
Development



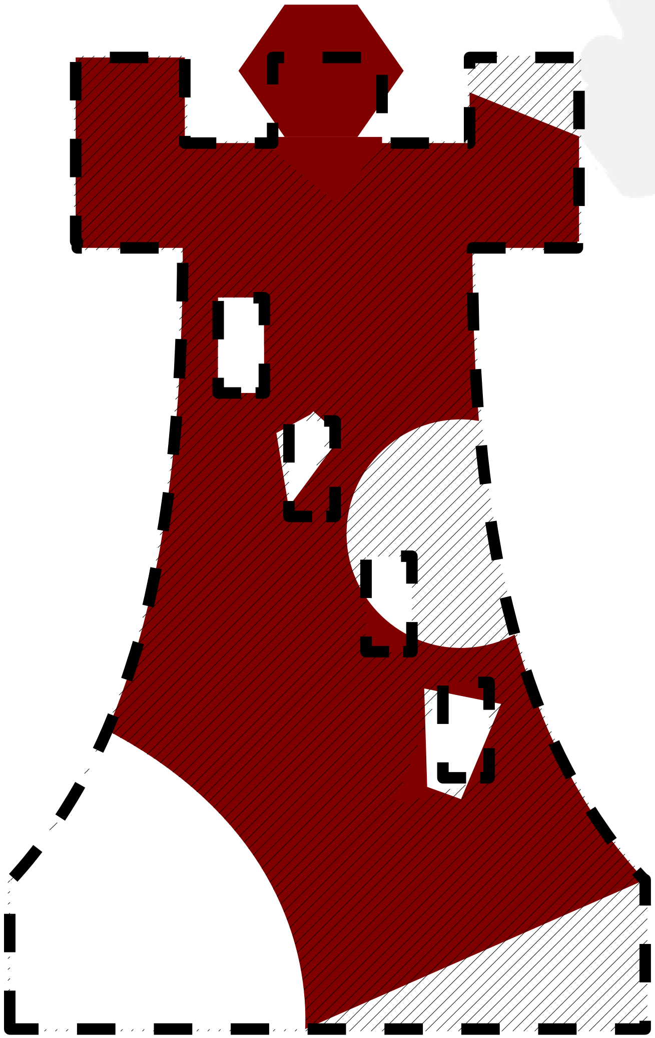
Architectural Issue Discovered



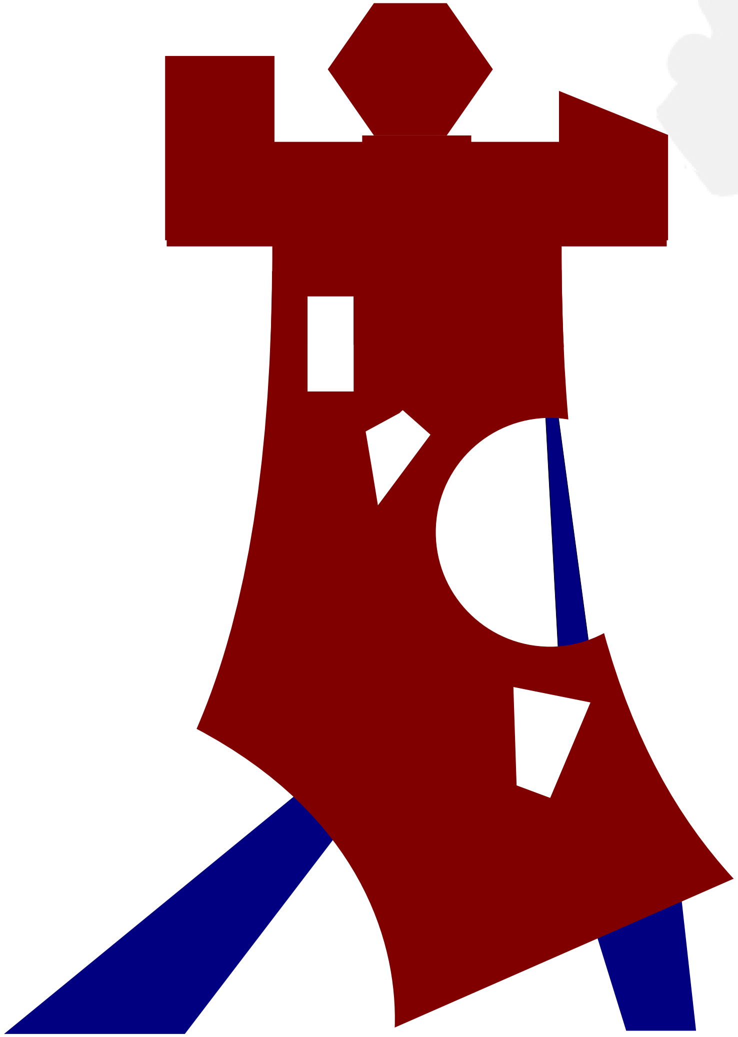
Development



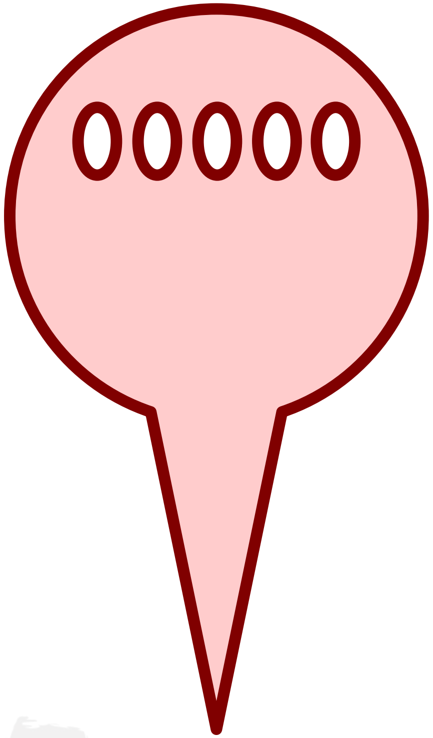
Development Finished



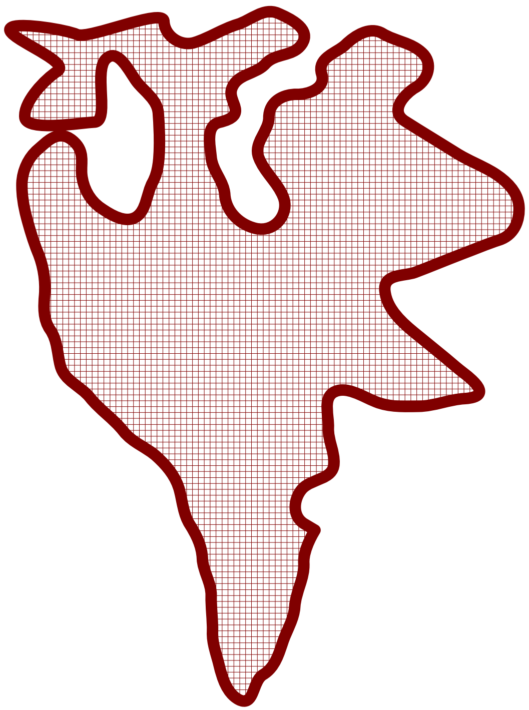
Delivery



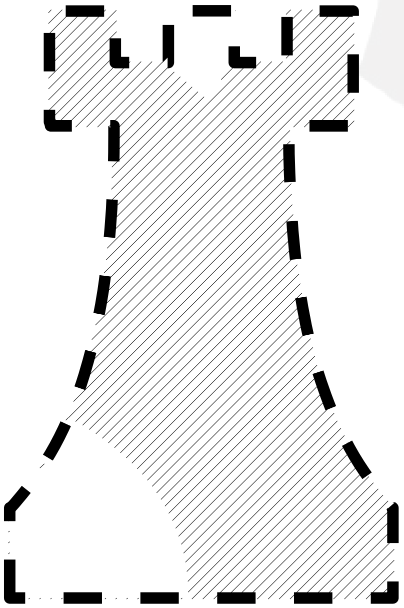
Morphing the System



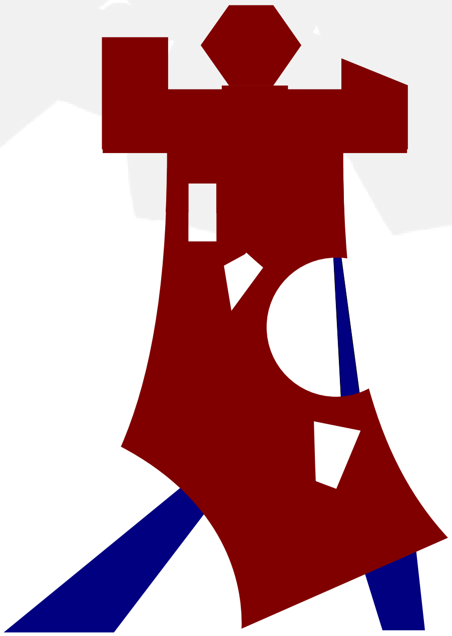
Desired



Described



Designed

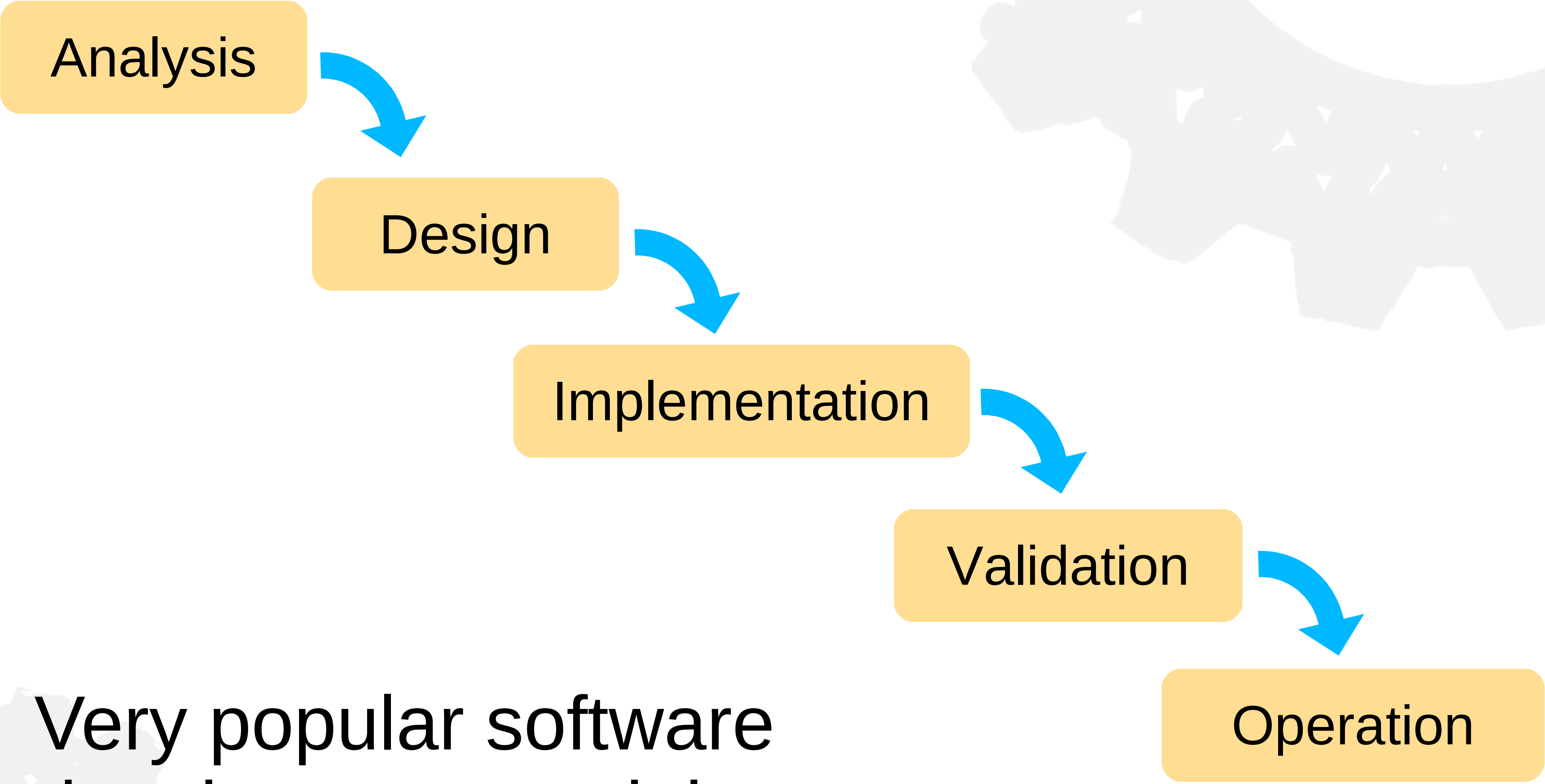


Deployed

We could do better than that

… in theory

Waterfall Model



Analysis

Design

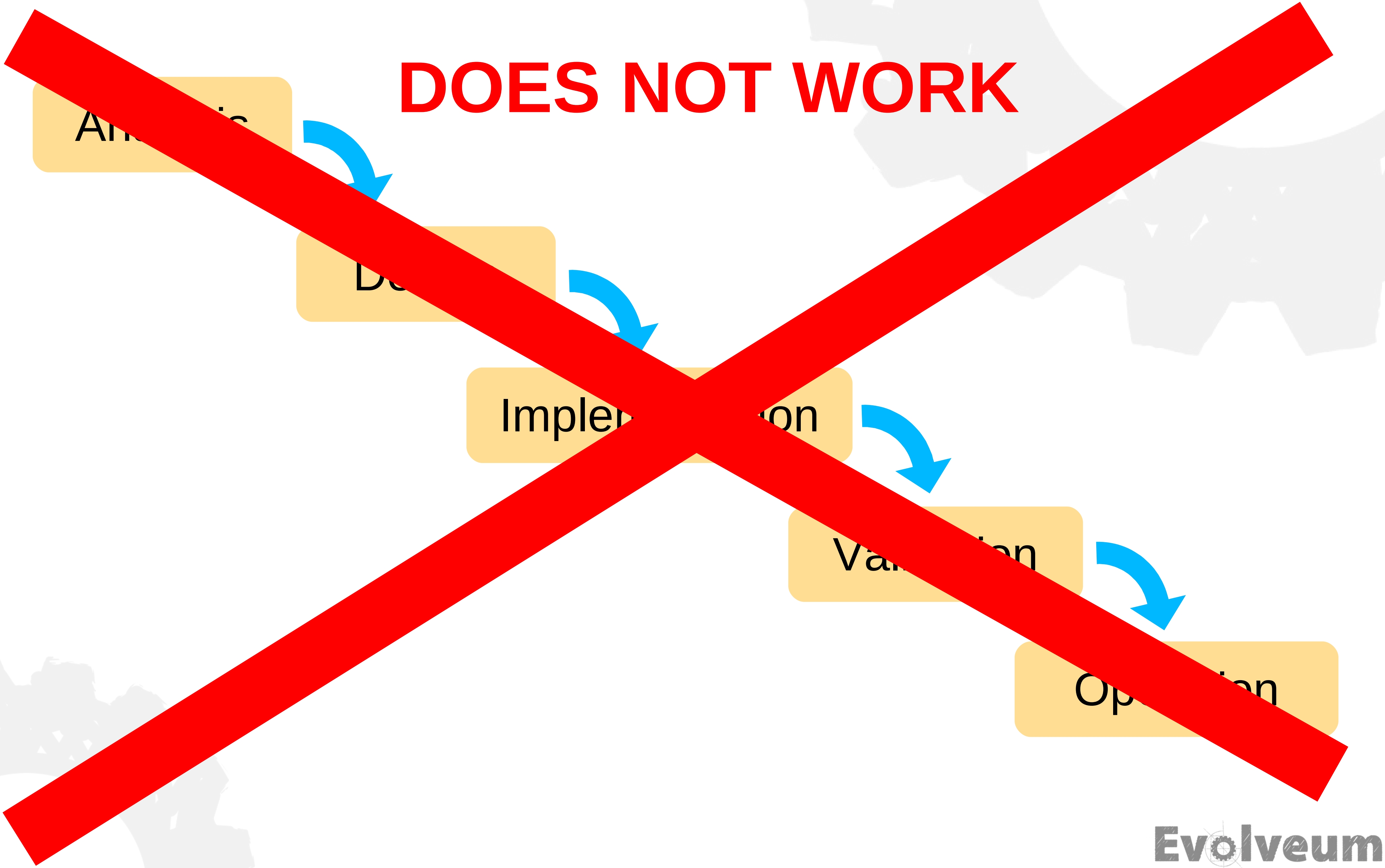
Implementation

Validation

Operation

Very popular software development model

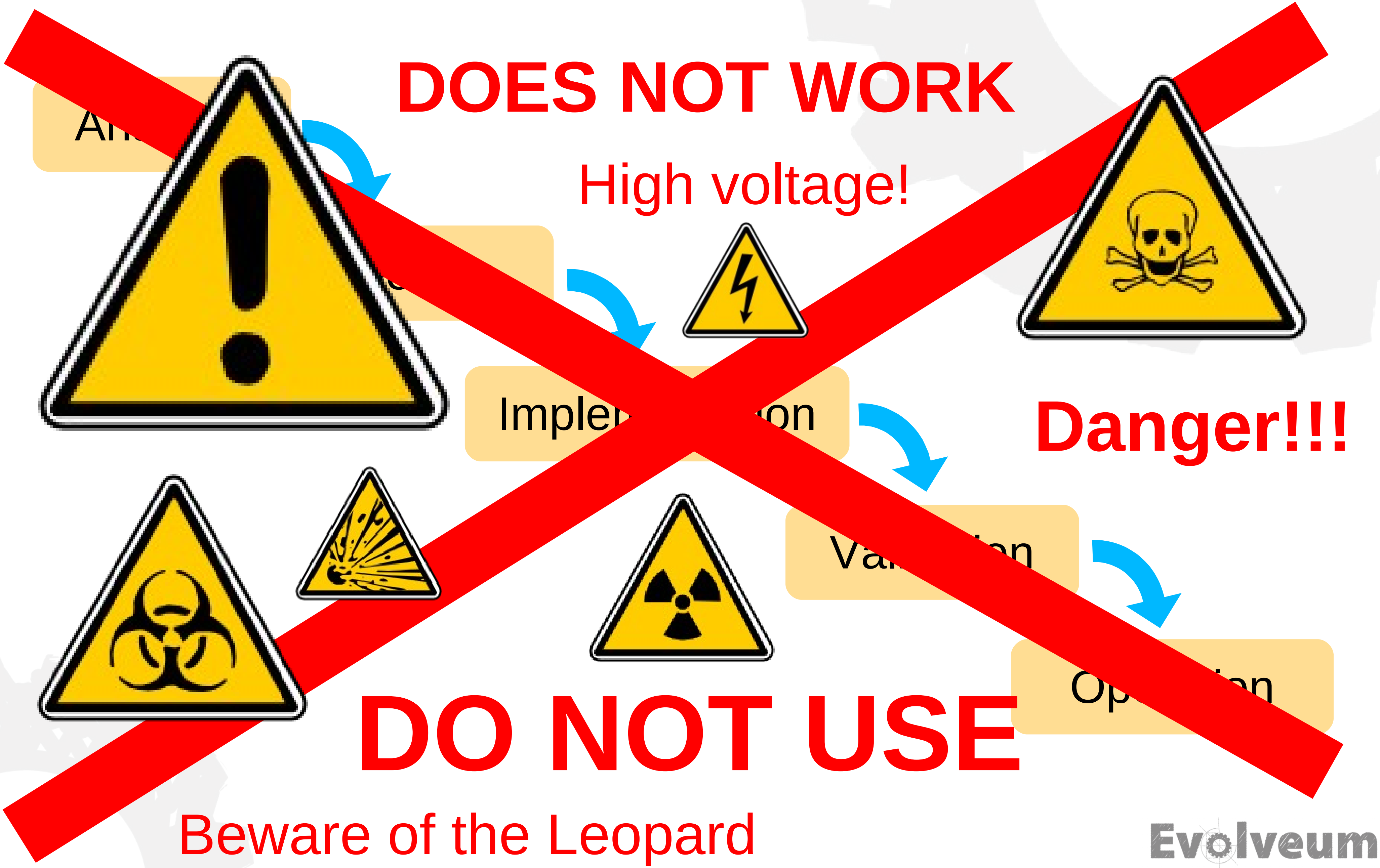
Waterfall Model



DOES NOT WORK

Waterfall Model

Hic sunt liones



DOES NOT WORK

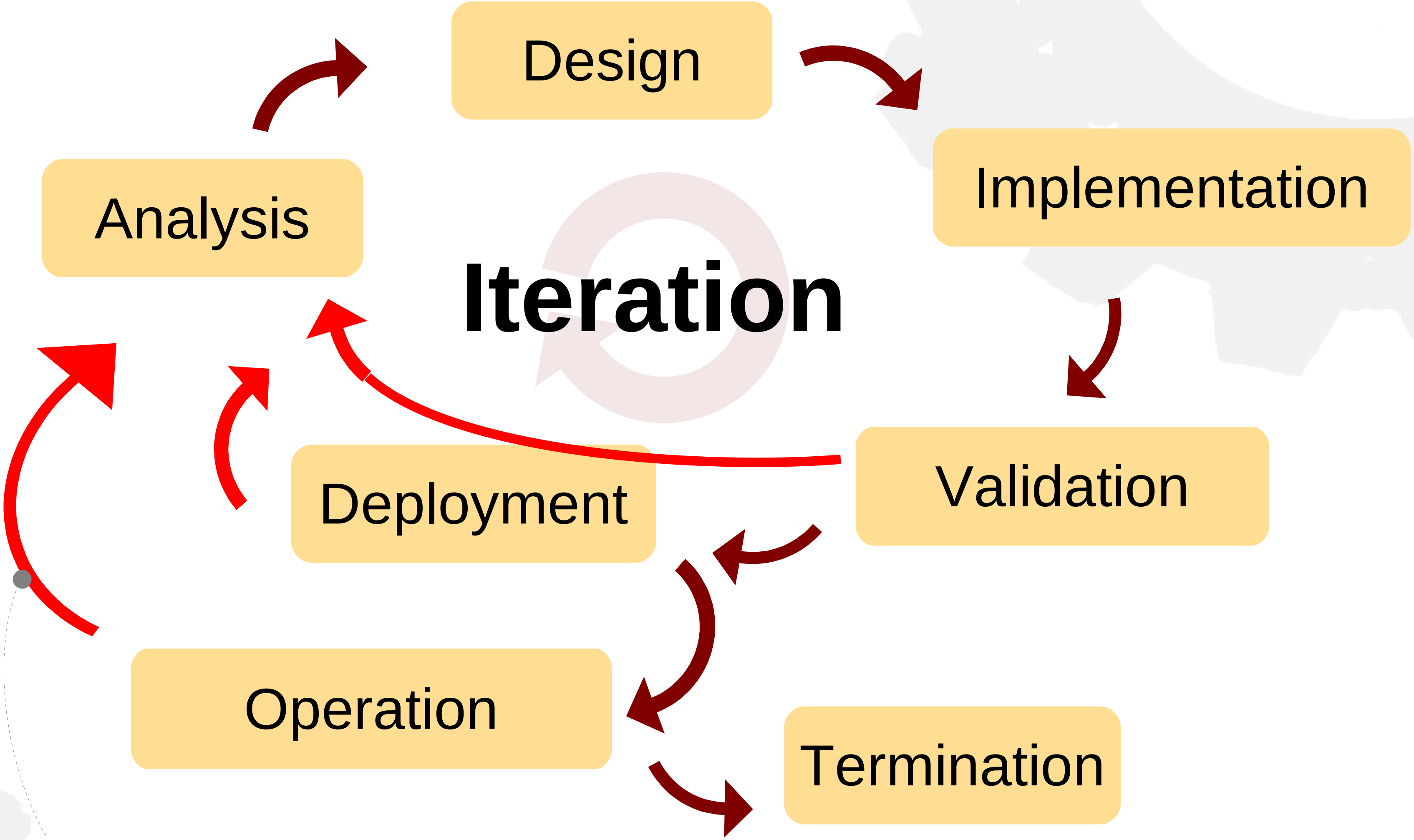
High voltage!

Danger!!!

DO NOT USE

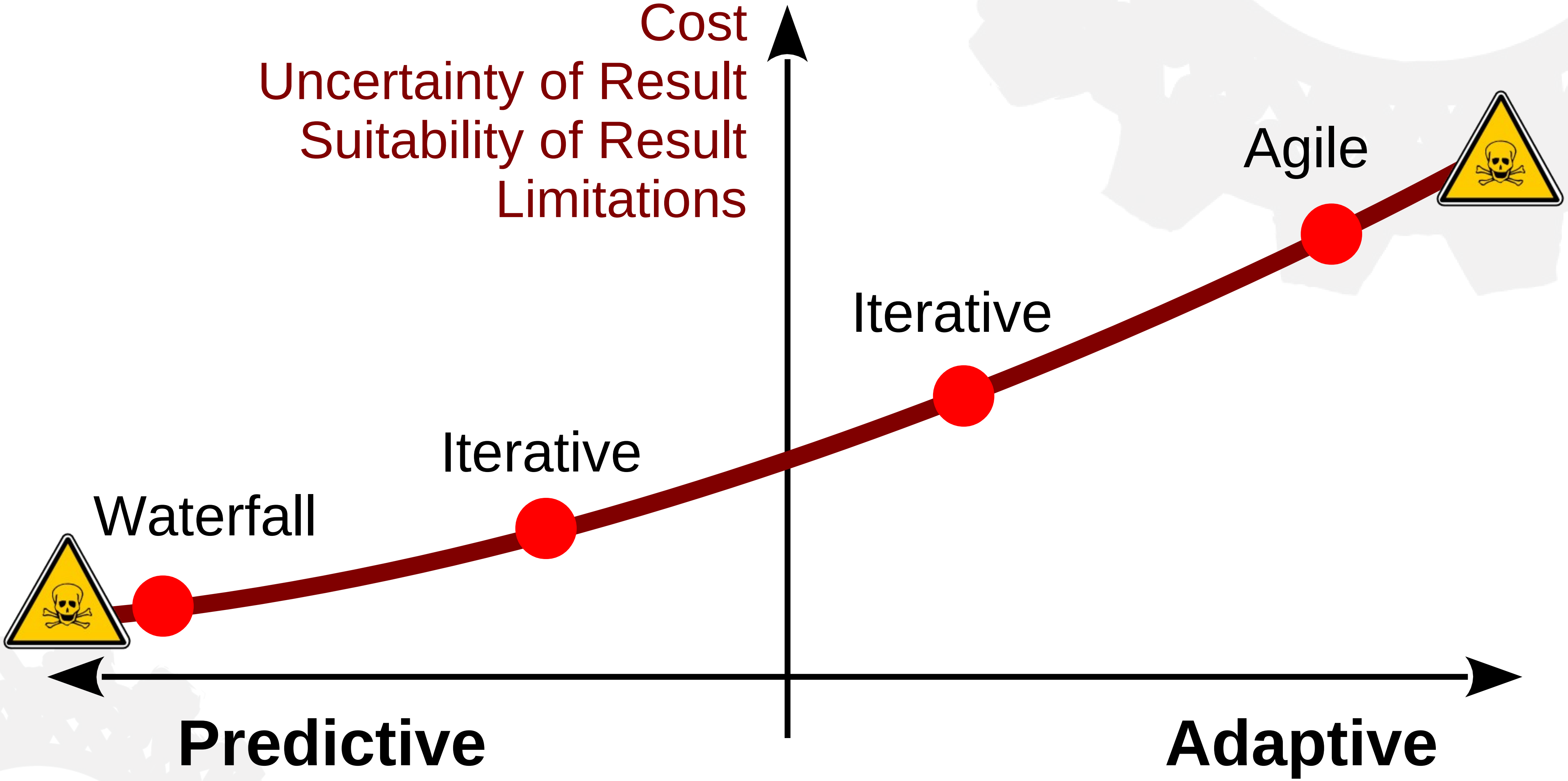
Beware of the Leopard

Iterative Development



- Feedback
  - Use knowledge gained in previous iteration

Development Methods Summary

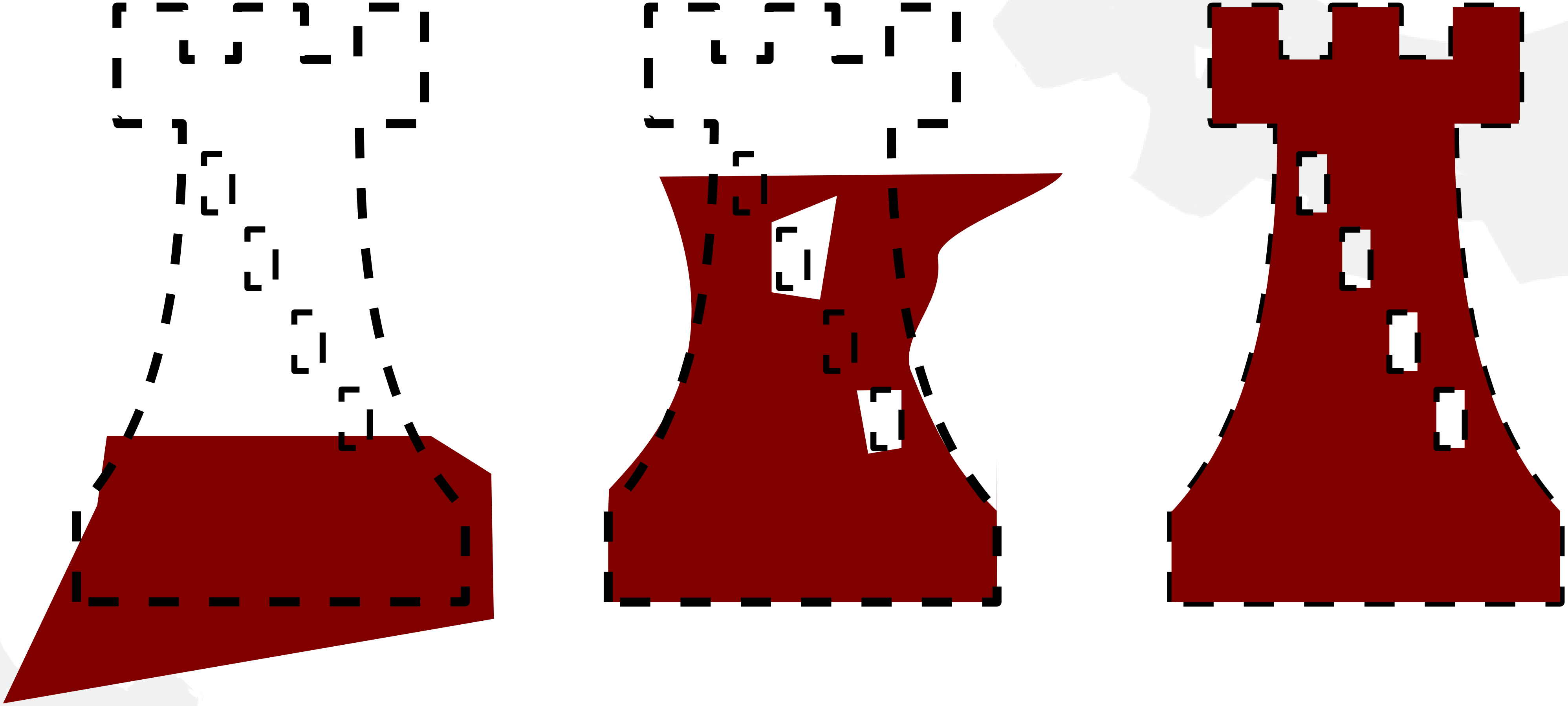


Evolveum

We could do better

… even in practice

Iterations and Increments



Software Developement
… in practice

- Do not try to design/implement everything
  - **Waterfall does not work!**
- Iterations and increments
  - But you need to have some idea about the desired result
- Beware the limitations
  - One size does not fit all
  - Agile does not **always** work
  - Golden hammer (anti-pattern)

Vision

Evolveum

Architecture and design

… in theory

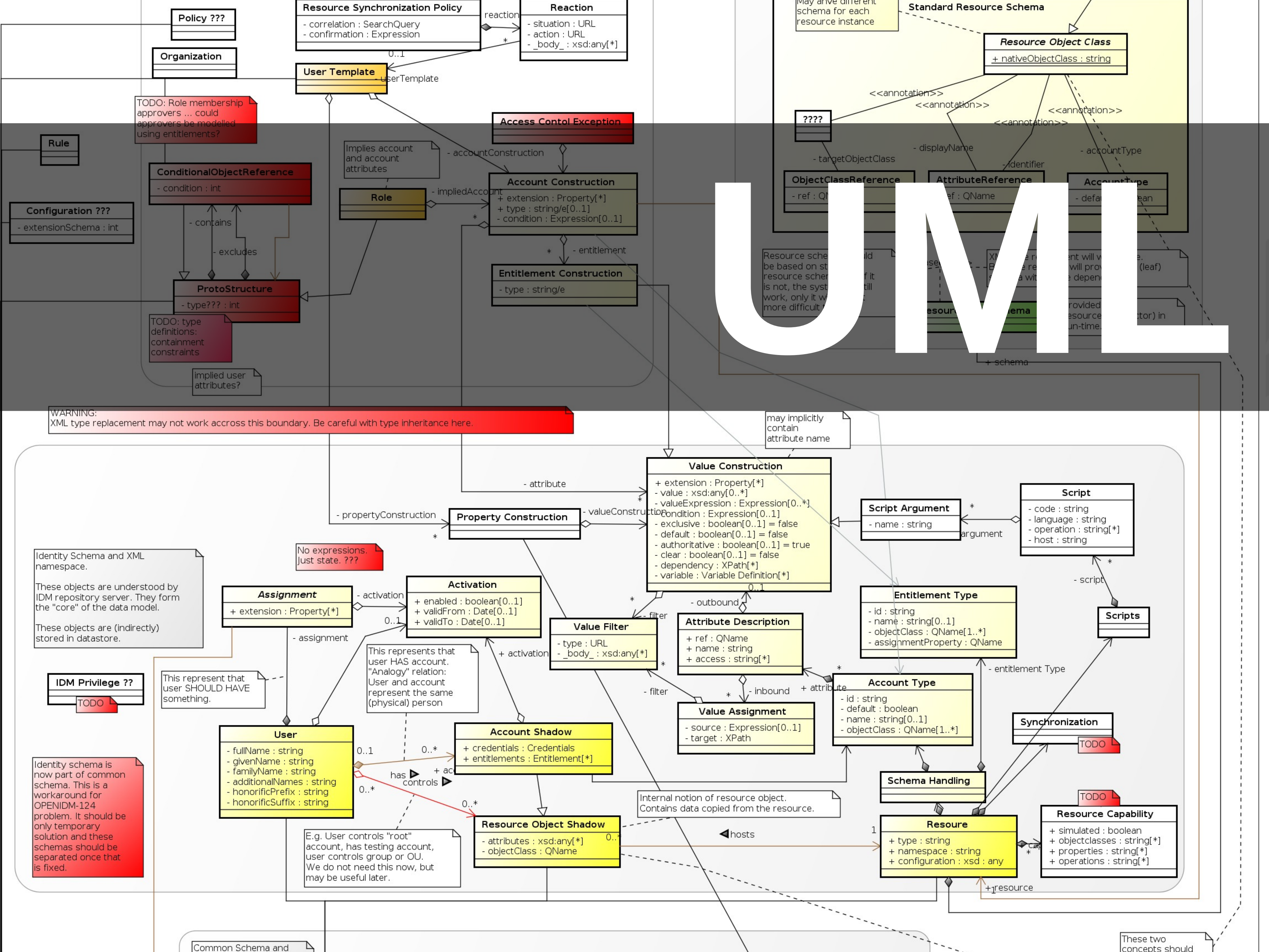
Model

Simplified

= imprecise

Overview

= better “handling”


Policy ???
Resource Synchronization Policy
- correlation : SearchQuery
- confirmation : Expression
reaction
Reaction
- situation : URL
- action : URL
- _body_ : xsd:any[*]
Organization
User Template
userTemplate
TODO: Role membership approvers ... could approvers be modelled using entitlements?
Rule
ConditionalObjectReference
- condition : int
Configuration ???
- extensionSchema : int
- contains
- excludes
ProtoStructure
- type??? : int
TODO: type definitions: containment constraints
implied user attributes?
Implies account and account attributes
Role
- impliedAccount
Access Contol Exception
- accountConstruction
Account Construction
+ extension : Property[*]
+ type : string/e[0..1]
- condition : Expression[0..1]
- entitlement
Entitlement Construction
- type : string/e
May ahve different schema for each resource instance
Standard Resource Schema
Resource Object Class
+ nativeObjectClass : string
<<annotation>>
????
- targetObjectClass
- displayName
- identifier
- accountType
ObjectClassReference
- ref : QName
AttributeReference
- ref : QName
AccountType
+ schema
WARNING:
XML type replacement may not work accross this boundary. Be careful with type inheritance here.
may implicitly contain attribute name
Value Construction
+ extension : Property[*]
- value : xsd:any[0..*]
- valueExpression : Expression[0..*]
- condition : Expression[0..1]
- exclusive : boolean[0..1] = false
- default : boolean[0..1] = false
- authoritative : boolean[0..1] = true
- clear : boolean[0..1] = false
- dependency : XPath[*]
- variable : Variable Definition[*]
- attribute
- propertyConstruction
Property Construction
- valueConstruction
Script Argument
- name : string
argument
Script
- code : string
- language : string
- operation : string[*]
- host : string
- script
Scripts
Identity Schema and XML namespace.
These objects are understood by IDM repository server. They form the "core" of the data model.
These objects are (indirectly) stored in datastore.
No expressions. Just state. ???
Assignment
+ extension : Property[*]
- activation
Activation
+ enabled : boolean[0..1]
+ validFrom : Date[0..1]
+ validTo : Date[0..1]
- assignment
+ activation
Value Filter
- type : URL
- _body_ : xsd:any[*]
- filter
- outbound
- inbound
Attribute Description
+ ref : QName
+ name : string
+ access : string[*]
Value Assignment
- source : Expression[0..1]
- target : XPath
Entitlement Type
- id : string
- name : string[0..1]
- objectClass : QName[1..*]
- assignmentProperty : QName
- entitlement Type
+ attribute
Account Type
- id : string
- default : boolean
- name : string[0..1]
- objectClass : QName[1..*]
IDM Privilege ??
TODO
This represent that user SHOULD HAVE something.
This represents that user HAS account. "Analogy" relation: User and account represent the same (physical) person
User
- fullName : string
- givenName : string
- familyName : string
- additionalNames : string
- honorificPrefix : string
- honorificSuffix : string
Account Shadow
+ credentials : Credentials
+ entitlements : Entitlement[*]
has
controls
Synchronization
Schema Handling
Identity schema is now part of common schema. This is a workaround for OPENIDM-124 problem. It should be only temporary solution and these schemas should be separated once that is fixed.
Resource Object Shadow
- attributes : xsd:any[*]
- objectClass : QName
E.g. User controls "root" account, has testing account, user controls group or OU. We do not need this now, but may be useful later.
Internal notion of resource object. Contains data copied from the resource.
hosts
Resoure
+ type : string
+ namespace : string
+ configuration : xsd : any
+resource
Resource Capability
+ simulated : boolean
+ objectclasses : string[*]
+ properties : string[*]
+ operations : string[*]
Common Schema and
These two concepts should

UML

Architecture and design

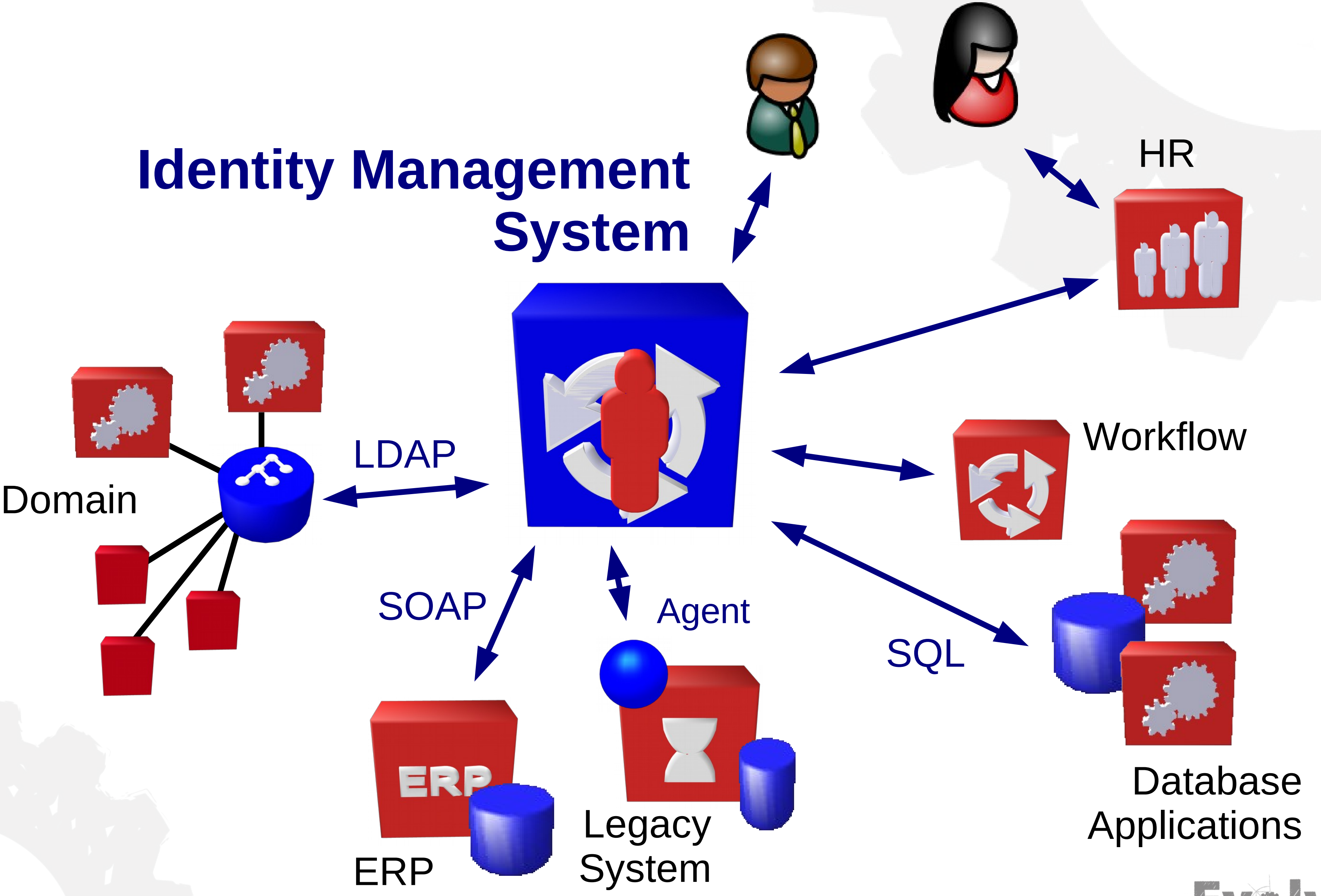
… in practice

Models

- Models in pure form (e.g. pure UML)
  - Limited usefulness
  - Fighting with tools instead of making progress
- Hybrid (customized) models
  - Very useful, especially in early phases
  - Difficult to maintain
- Free-form diagrams
  - Whiteboard – absolutely necessary
  - Brainstorming, early “validation”

Informal Architecture Diagram

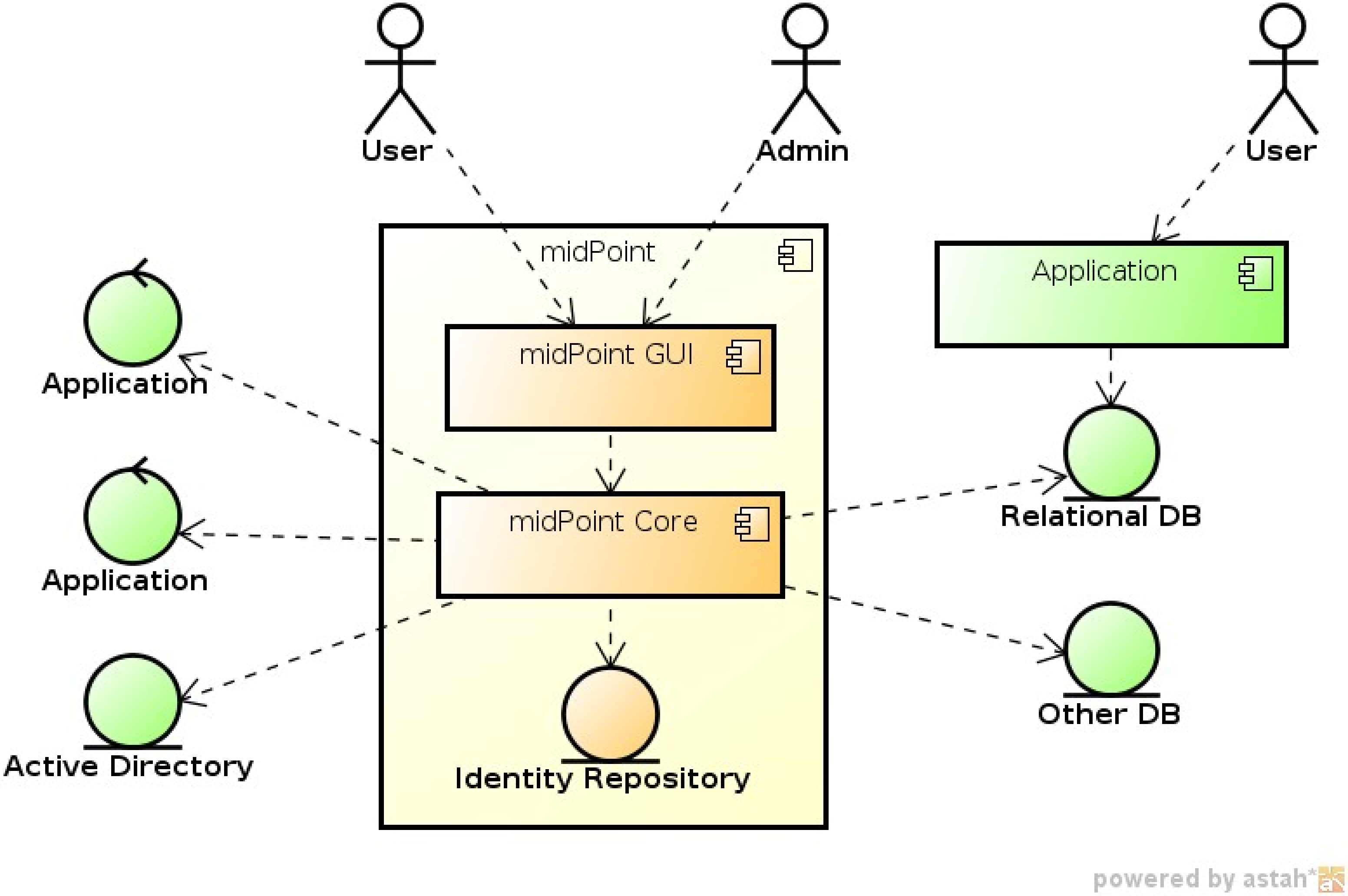
(Technical marketing)



Evolveum

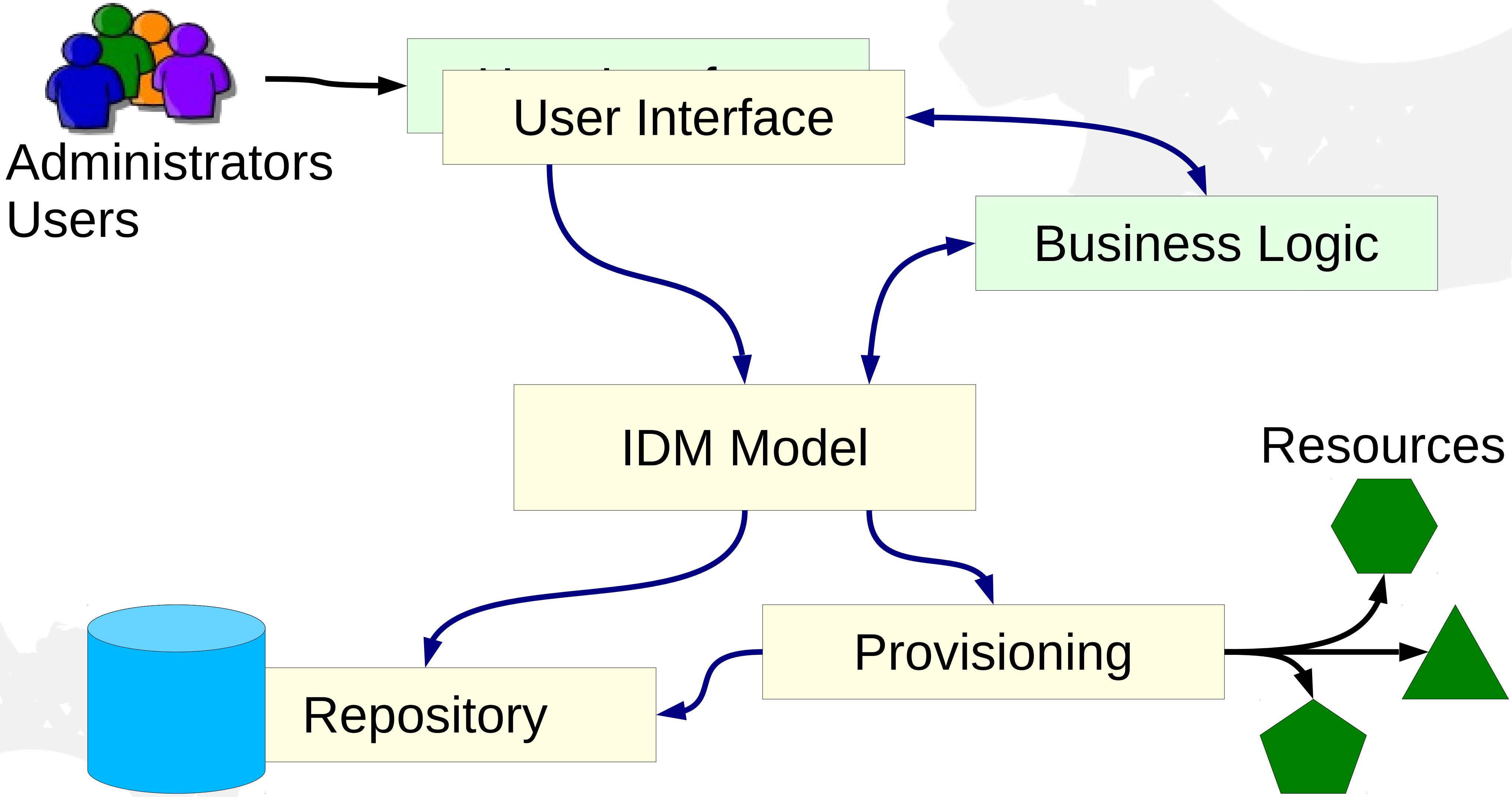
Formal Architecture Diagram

(I have UML and I'm not afraid to use it)



Informal Component Diagram

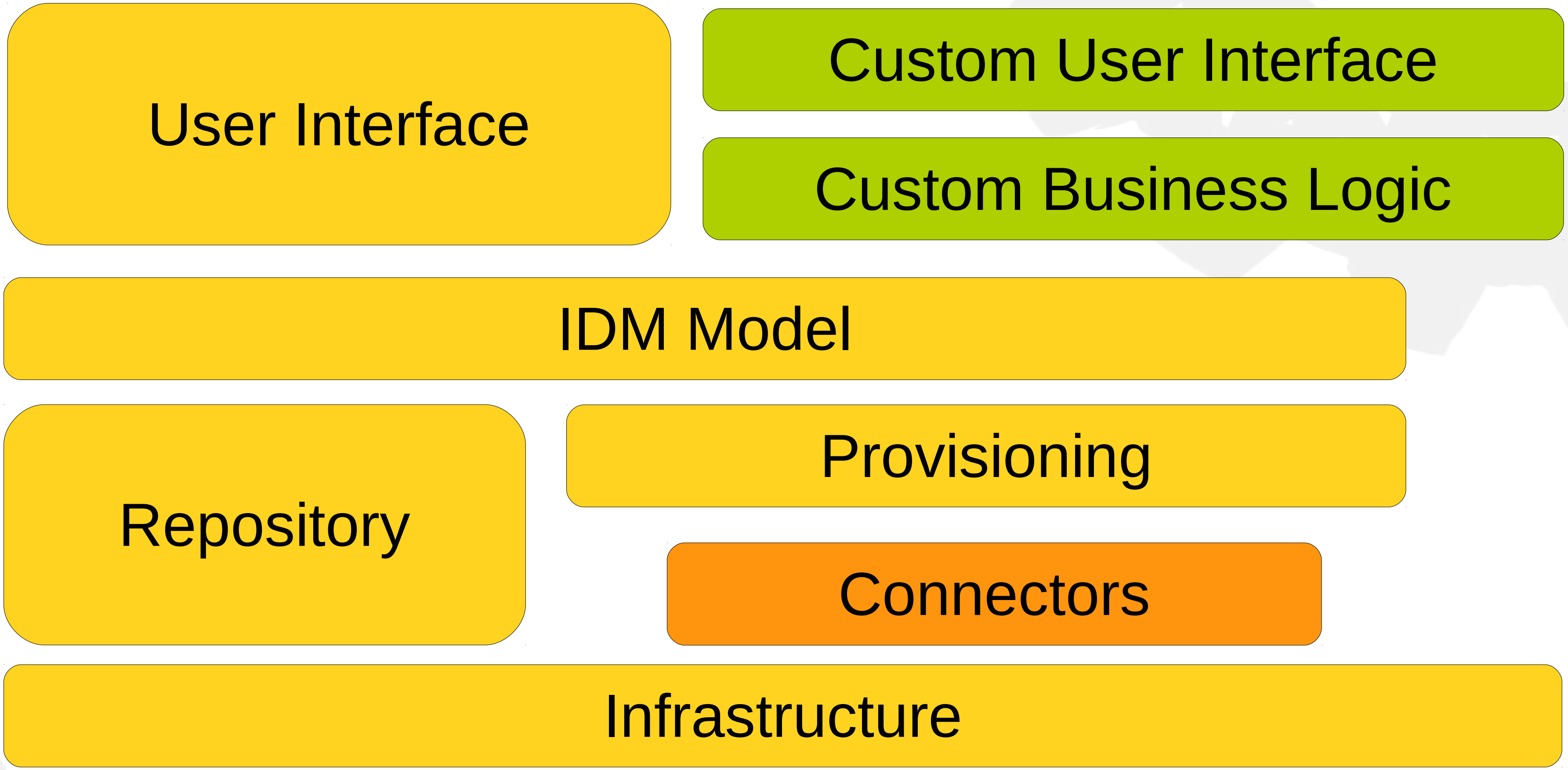
(Whiteboard 2.0)



Evolveum

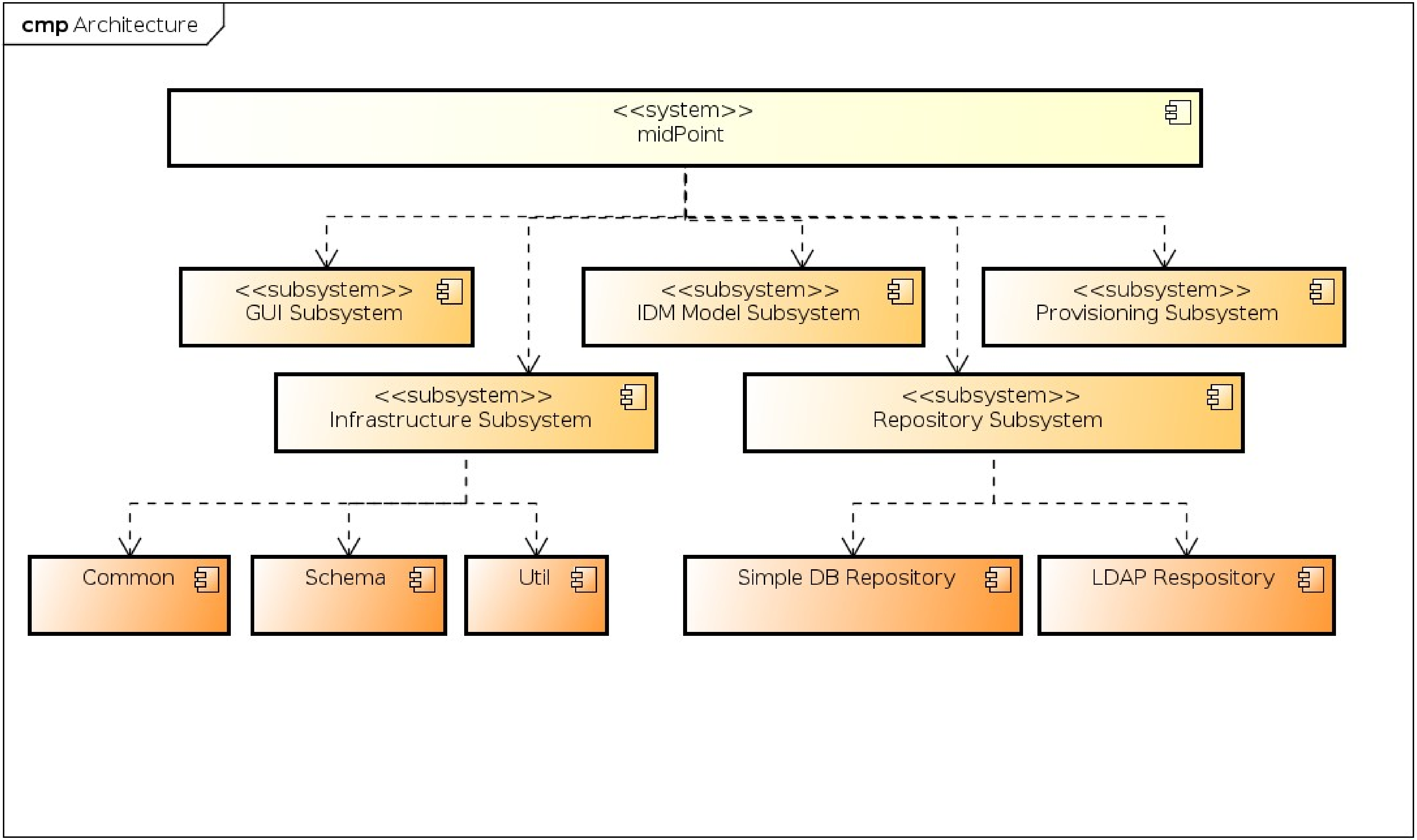
Marketing-Oriented Diagram

(Boxes and more boxes)

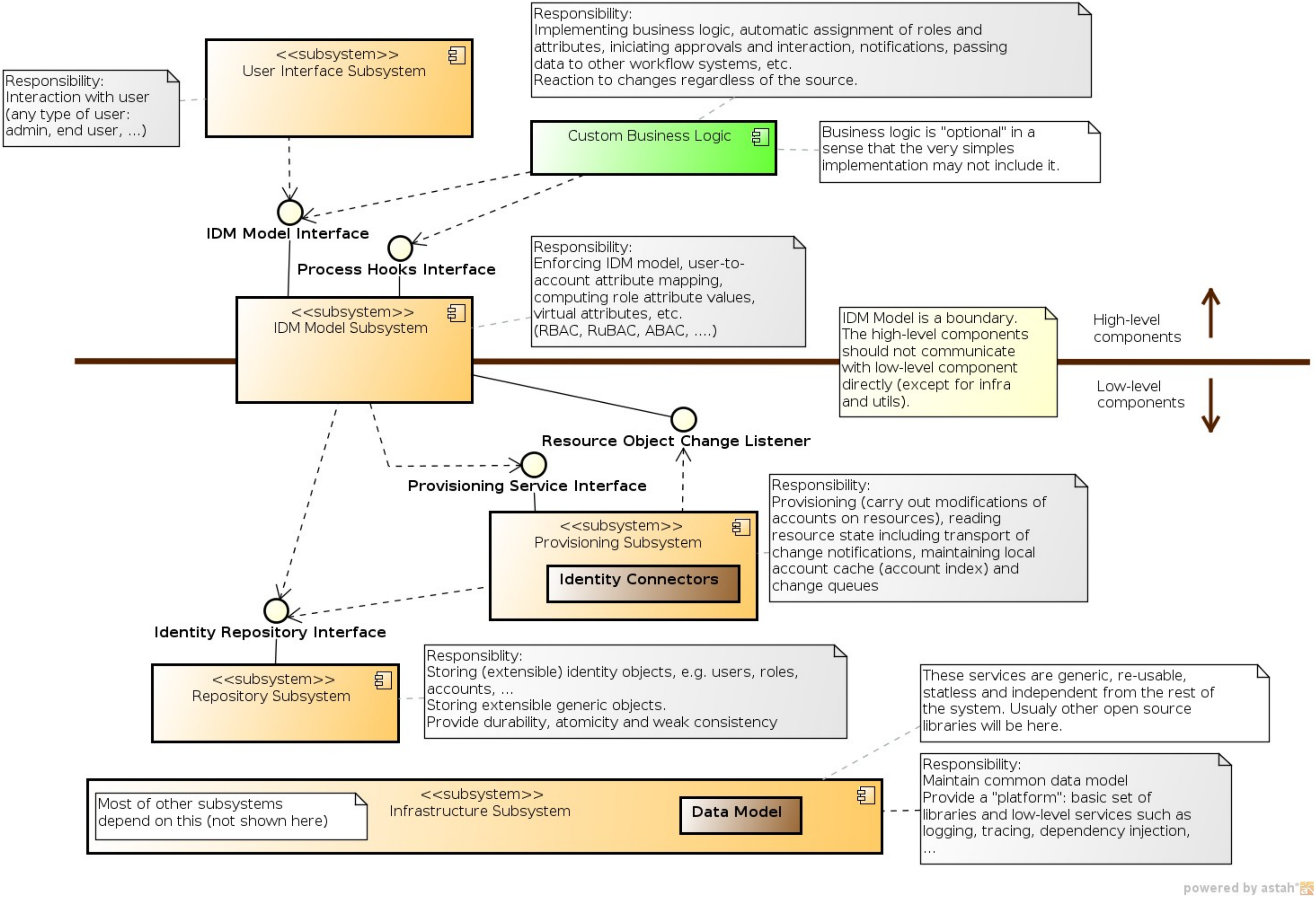


If you ever see this: run away!

System Decomposition



Modular and Component Structure



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 and utils).

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

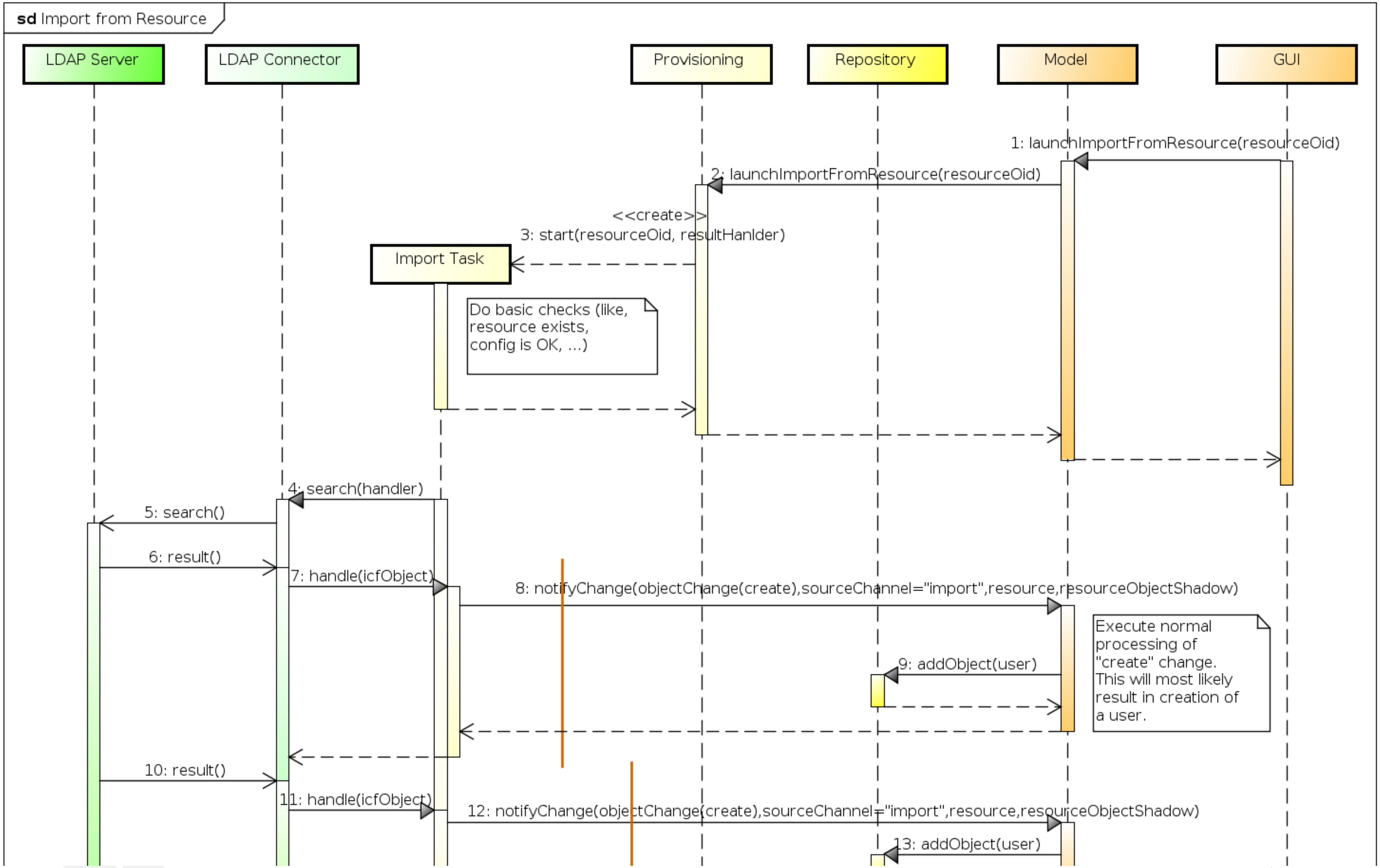
Responsiblity:
Storing (extensible) identity objects, e.g. users, roles, accounts, ...
Storing extensible generic objects.
Provide durability, atomicity and weak consistency

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, ...

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

Component Interactions



sd Import from Resource

LDAP Server

LDAP Connector

Provisioning

Repository

Model

GUI

1: launchImportFromResource(resourceOid)

2: launchImportFromResource(resourceOid)

<<create>>

3: start(resourceOid, resultHanlder)

Import Task

Do basic checks (like, resource exists, config is OK, ...)

4: search(handler)

5: search()

6: result()

7: handle(icfObject)

8: notifyChange(objectChange(create),sourceChannel="import",resource,resourceObjectShadow)

Execute normal processing of "create" change. This will most likely result in creation of a user.

9: addObject(user)

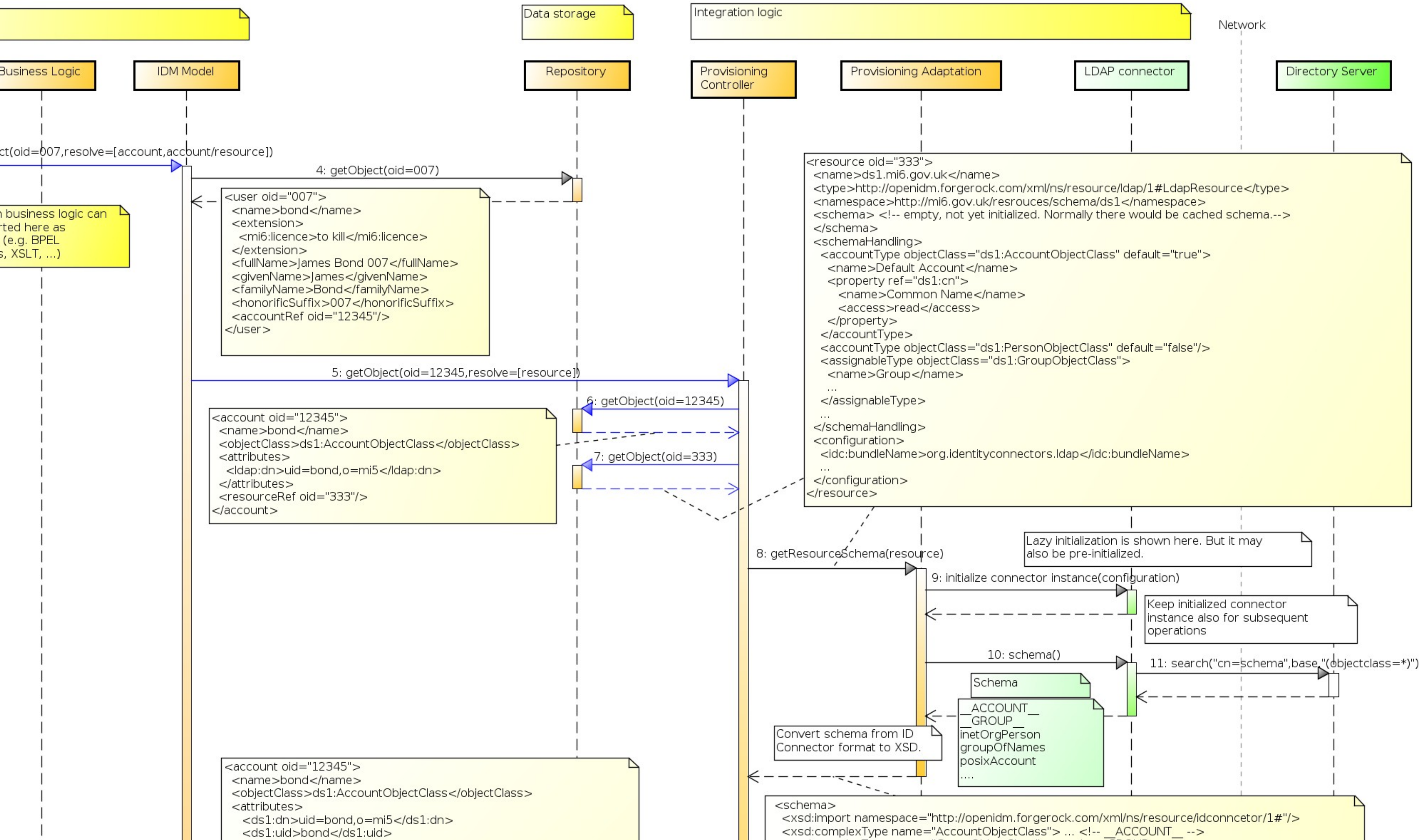
10: result()

11: handle(icfObject)

12: notifyChange(objectChange(create),sourceChannel="import",resource,resourceObjectShadow)

13: addObject(user)

Component Interactions



n business logic can
rted here as
(e.g. BPEL
s, XSLT, ...)

```
<user oid="007">
  <name>bond</name>
  <extension>
    <mi6:licence>to kill</mi6:licence>
  </extension>
  <fullName>James Bond 007</fullName>
  <givenName>James</givenName>
  <familyName>Bond</familyName>
  <honorificSuffix>007</honorificSuffix>
  <accountRef oid="12345"/>
</user>
```

```
<account oid="12345">
  <name>bond</name>
  <objectClass>ds1:AccountObjectClass</objectClass>
  <attributes>
    <ldap:dn>uid=bond,o=mi5</ldap:dn>
  </attributes>
  <resourceRef oid="333"/>
</account>
```

```
<resource oid="333">
  <name>ds1.mi6.gov.uk</name>
  <type>http://openidm.forgerock.com/xml/ns/resource/ldap/1#LdapResource</type>
  <namespace>http://mi6.gov.uk/resrouces/schema/ds1</namespace>
  <schema> <!-- empty, not yet initialized. Normally there would be cached schema.-->
  </schema>
  <schemaHandling>
    <accountType objectClass="ds1:AccountObjectClass" default="true">
      <name>Default Account</name>
      <property ref="ds1:cn">
        <name>Common Name</name>
        <access>read</access>
      </property>
    </accountType>
    <accountType objectClass="ds1:PersonObjectClass" default="false"/>
    <assignableType objectClass="ds1:GroupObjectClass">
      <name>Group</name>
      ...
    </assignableType>
    ...
  </schemaHandling>
  <configuration>
    <idc:bundleName>org.identityconnectors.ldap</idc:bundleName>
    ...
  </configuration>
</resource>
```

Lazy initialization is shown here. But it may also be pre-initialized.

Keep initialized connector instance also for subsequent operations

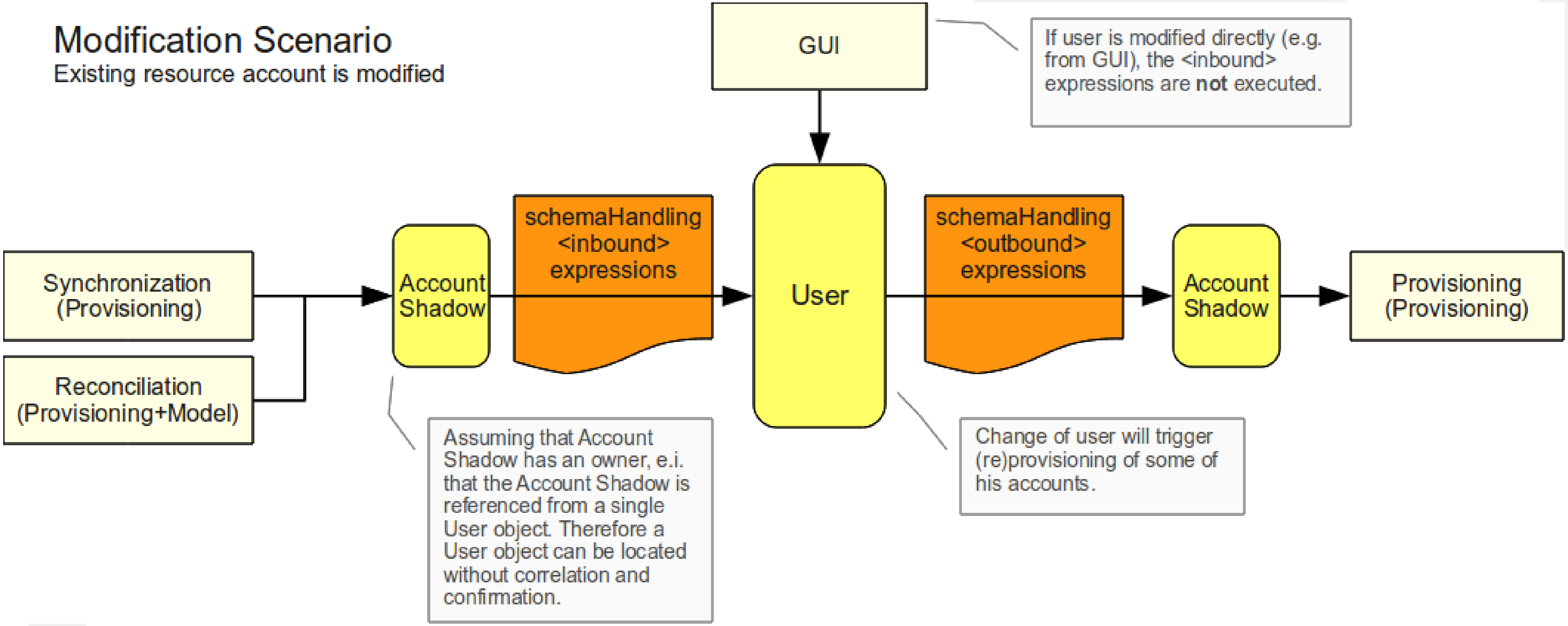
```
__ACCOUNT__
__GROUP__
inetOrgPerson
groupOfNames
posixAccount
....
```

Convert schema from ID Connector format to XSD.

```
<account oid="12345">
  <name>bond</name>
  <objectClass>ds1:AccountObjectClass</objectClass>
  <attributes>
    <ds1:dn>uid=bond,o=mi5</ds1:dn>
    <ds1:uid>bond</ds1:uid>
```

```
<schema>
  <xsd:import namespace="http://openidm.forgerock.com/xml/ns/resource/idconncetor/1#"/>
  <xsd:complexType name="AccountObjectClass"> ... <!-- __ACCOUNT__ -->
```

Component Interactions



Modification Scenario
Existing resource account is modified

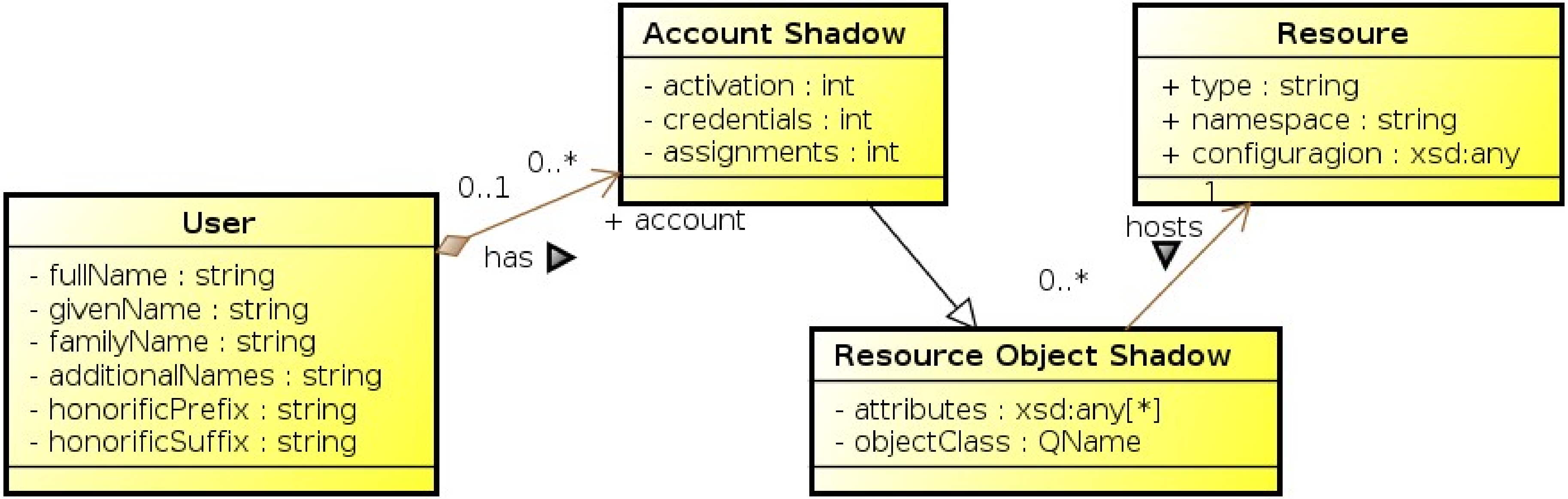
GUI
Synchronization (Provisioning)
Reconciliation (Provisioning+Model)
Account Shadow
schemaHandling <inbound> expressions
User
schemaHandling <outbound> expressions
Account Shadow
Provisioning (Provisioning)

If user is modified directly (e.g. from GUI), the <inbound> expressions are **not** executed.

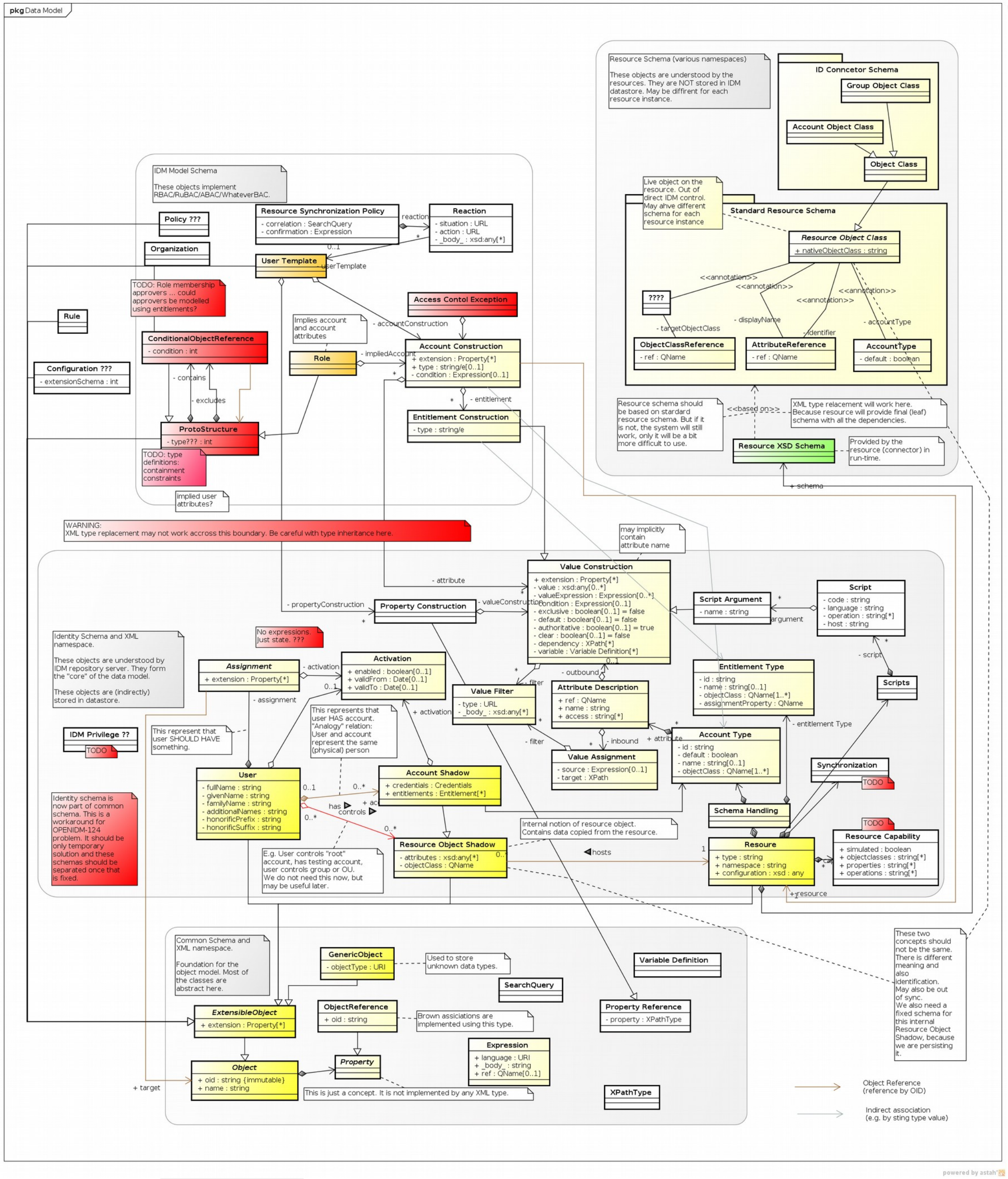
Assuming that Account Shadow has an owner, e.i. that the Account Shadow is referenced from a single User object. Therefore a User object can be located without correlation and confirmation.

Change of user will trigger (re)provisioning of some of his accounts.

Data Structures



Complex Data Structures



IDM Model Schema

These objects implement RBAC/RuBAC/ABAC/WhateverBAC.

TODO: Role membership approvers ... could approvers be modelled using entitlements?

Implies account and account attributes

TODO: type definitions: containment constraints

implied user attributes?

Resource Schema (various namespaces)

These objects are understood by the resources. They are NOT stored in IDM datastore. May be diffirent for each resource instance.

Live object on the resource. Out of direct IDM control. May ahve different schema for each resource instance

Resource schema should be based on stardard resource schema. But if it is not, the system will still work, only it will be a bit more difficult to use.

XML type relacement will work here. Because resource will provide final (leaf) schema with all the dependencies.

Provided by the resource (connector) in run-time.

WARNING:
XML type replacement may not work accross this boundary. Be careful with type inheritance here.

may implicitly contain attribute name

Identity Schema and XML namespace.

These objects are understood by IDM repository server. They form the "core" of the data model.

These objects are (indirectly) stored in datastore.

No expressions. Just state. ???

This represent that user SHOULD HAVE something.

This represents that user HAS account. "Analogy" relation: User and account represent the same (physical) person

Identity schema is now part of common schema. This is a workaround for OPENIDM-124 problem. It should be only temporary solution and these schemas should be separated once that is fixed.

E.g. User controls "root" account, has testing account, user controls group or OU. We do not need this now, but may be useful later.

Internal notion of resource object. Contains data copied from the resource.

Common Schema and XML namespace.

Foundation for the object model. Most of the classes are abstract here.

Used to store unknown data types.

Brown assiciations are implemented using this type.

This is just a concept. It is not implemented by any XML type.

These two concepts should not be the same. There is different meaning and also identification. May also be out of sync. We also need a fixed schema for this internal Resource Object Shadow, because we are persisting it.

Object Reference (reference by OID)

Indirect association (e.g. by sting type value)

- Hard to maintain
- Data schema
- Generate?

Architecture Model Summary

- Operates with concepts
  - May or may not map to final components, interfaces, ...
- Difficult to align with implementation
  - ... and not efficient to reach 100% alignment
  - The model should be guideline, not dogma
- Model ≠ Architecture
  - Architecture is much more:
    - Textual descriptions, explanations, description of concepts
    - Motivations, design decisions, trade-offs, future expectations
  - **Beware** of tools that promise to simplify that

WARNING

Architectural Principles

Those are (very) useful

- Separation of concerns
- Dependency inversion principle
- Acyclic dependencies principle
- Stable abstractions principle
- Stable dependencies principle
- Open-closed principle
- Single responsibility principle
- Interface segregation principle
- …

Evolveum

When architecture goes wrong …

Fallacies, Antipatterns, Rot & Smell

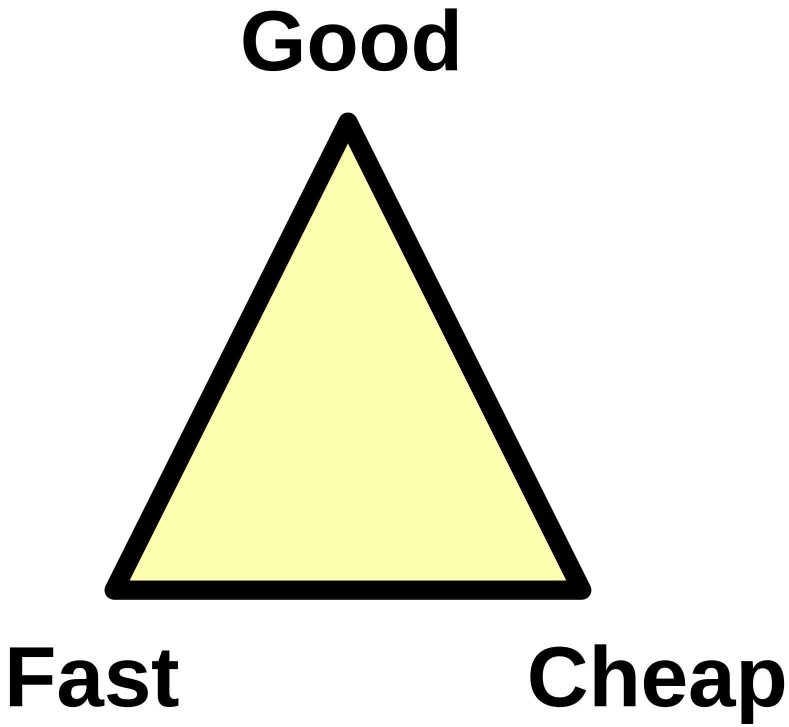
- Fallacies of distributed computing
  - Network is reliable, Latency is zero, Bandwidth is infinite, ...
- Architectural antipatterns
  - Big ball of mud, Design by committee, Not invented here, ...
- Symptoms of rotting design
  - Rigidity, Fragility, Immobility, Viscosity
- Code smell
  - Duplicated code, Contrived complexity, Feature envy, ...

Common Problems

- Too little analysis / design
  - Especially in agile and open source
- Too much architecture ("stratospheric architecture")
  - Pretty concepts that never get implemented
- No environment analysis
- Unmaintained architecture
  - Architect *did his work* at beginning of the project

    … and then left
  - Architecture is a mutable thing! Needs constant maintenance.

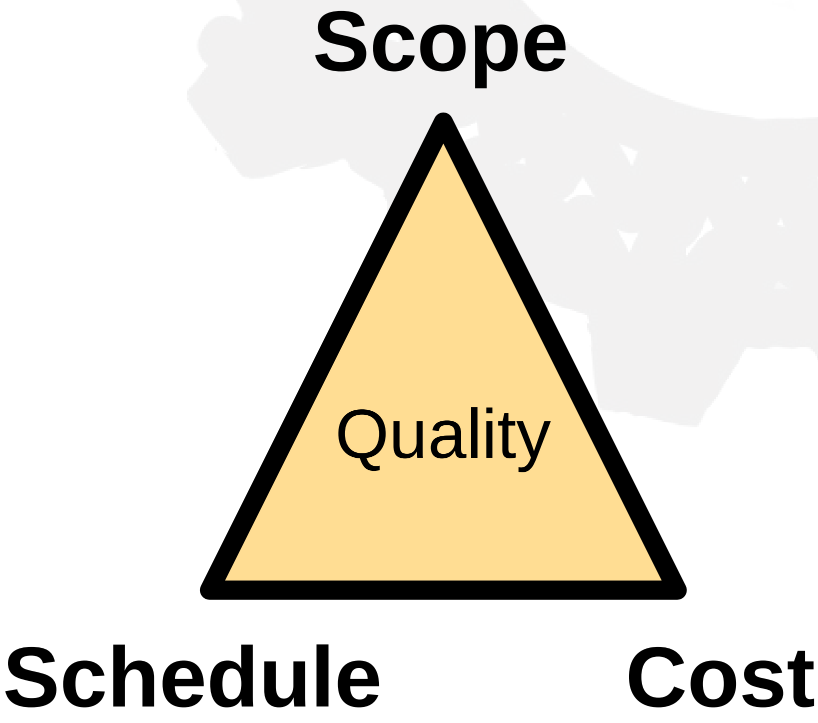
Iron Triangle



Good

Fast

Cheap



Scope

Quality

Schedule

Cost

Pick any two ...
... the third will follow

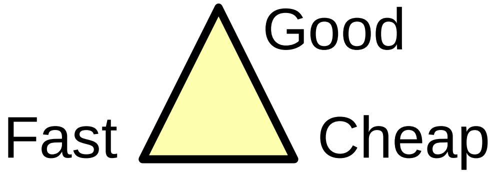
At least one corner must be variable, otherwise quality will suffer

Moving Target

- Requirements are incomplete and changing
- Environment is changing

  => software must change

- Architecture must be able to adapt
- Expect that you will have to make changes
- Do not forget about Iron Triangle

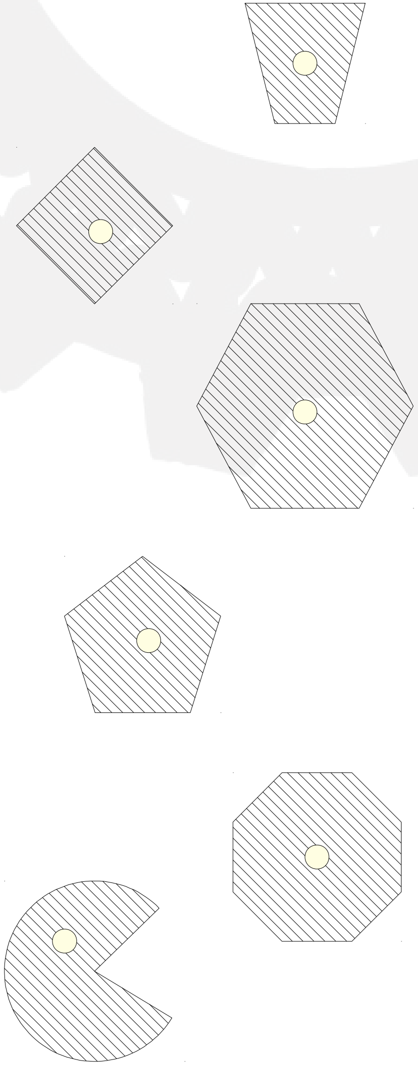


Buzzword-Oriented Architecture

- Very common approach
- Huge problem
- Solution: known what you are doing
  - Understand the technology before committing to it
- History repeating
  - Basic principles do not change often

History Repeating

- 1976: RFC 707
- 1981: Xerox Courier
- 1991: CORBA
- 1993: DCE/RPC → DCOM
- 1995: SunRPC
- 1998: SOAP
- 200x: “RESTful” API



What we can do?

Form follows purpose

Form Follows Purpose

versus

Pragmatic Approach

- Focus on the effects of the architecture
  - Emphasize the aspects that can help achieve results
  - Ignore aspects that does not influence result
- Common sense, simplicity
- Continuous change
- Skepticism
  - Continual testing, systematic doubt
  - True knowledge is uncertain

Questions and Answers

Thank You

Radovan Semančík

www.evolveum.com

Evolveum