SOFAS,
SOFTWARE ANALYSIS AS A SERVICE

Giacomo Ghezzi

University of Zurich UZH

Department of Informatics

s.e.a.l.
software evolution & architecture lab

RED QUEEN'S HYPOTHESIS

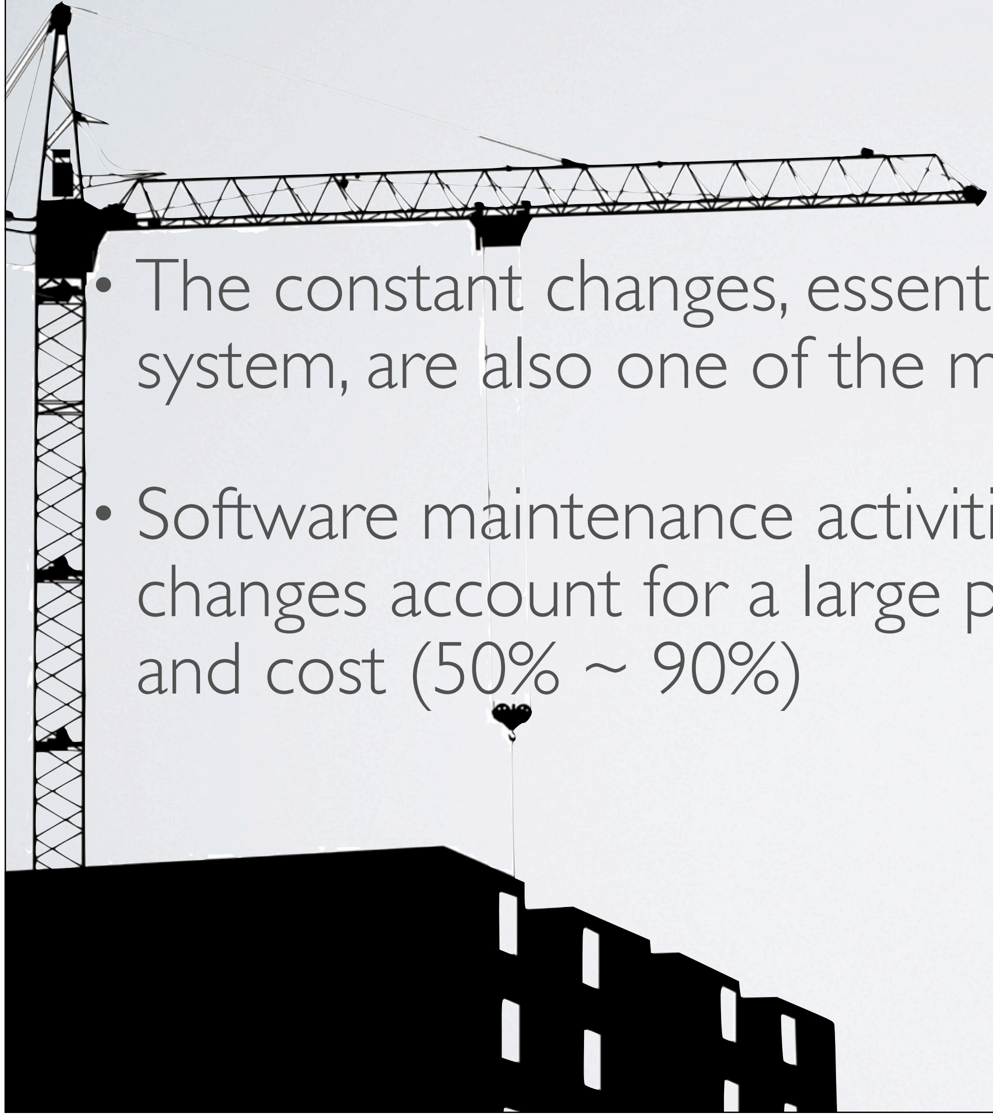
"It takes all the running you can do, to keep in the same place."

"Through the Looking Glass", L. Carroll

"For an evolutionary system, continuing development is needed just in order to maintain its fitness relative to the systems it is co-evolving with."

L. van Valen

THE CHANGE PROBLEM



- The constant changes, essential for the success of a software system, are also one of the main causes of its dismissal

- Software maintenance activities aimed at controlling these changes account for a large portion of the development effort and cost (50% ~ 90%)

SOFTWARE EVOLUTION ANALYSIS

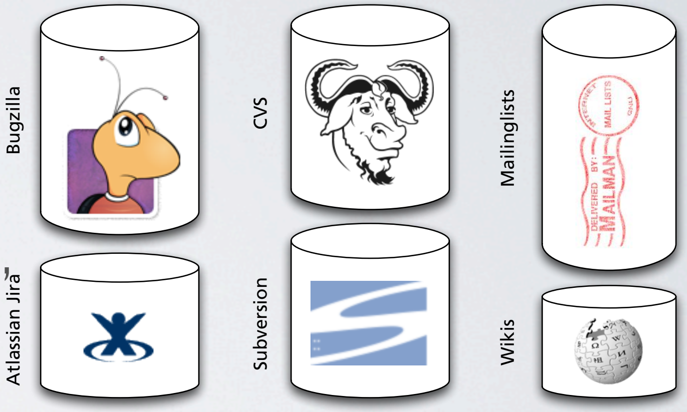
- Having an always up to date and thorough view of a software system, its health and history is key

- **Software repositories** can provide a huge amount of data useful for that purpose

SOFTWARE EVOLUTION ANALYSIS

- **Software repositories** were used primarily for historical records

- **Gold rush** to exploit this data to support software evolution, improve software design/reuse, understand and support development, etc.



- Mining software archives have come up for many types of analyses:
  - change types
  - change couplings
  - defect prediction
  - developer networks

WHERE'S THE PROBLEM?

WHERE'S THE PROBLEM?

Analyses are hard to access and use

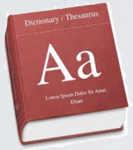
WHERE'S THE PROBLEM?

Analyses are hard to access and use

Analyses lack a clear and uniform data representation

WHERE'S THE PROBLEM?

Analyses are hard to access and use

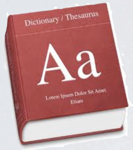


Analyses lack a clear and uniform data representation

No straight forward integration between different analyses exists.

WHERE'S THE PROBLEM?

Analyses are hard to access and use



Analyses lack a clear and uniform data representation

No straight forward integration between different analyses exists.

People keep reinventing the same wheels

THE NEED FOR A DIFFERENT APPROACH

THE NEED FOR A DIFFERENT APPROACH

1. How can we offer the functionalities of this plethora of evolution analyses in a consistent and efficient way?

THE NEED FOR A DIFFERENT APPROACH

1. How can we offer the functionalities of this plethora of evolution analyses in a consistent and efficient way?
2. How can we effectively describe the heterogeneous and wide-ranging data produced by such analyses?

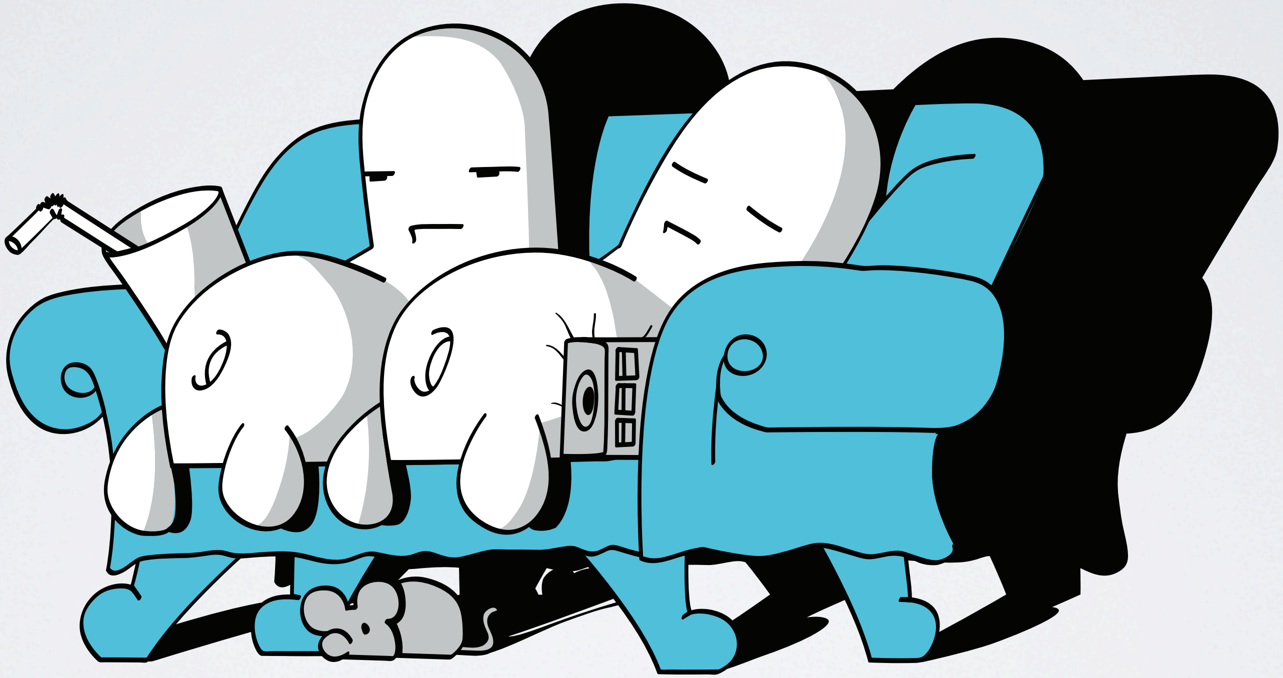
THE NEED FOR A DIFFERENT APPROACH

1. How can we offer the functionalities of this plethora of evolution analyses in a consistent and efficient way?
2. How can we effectively describe the heterogeneous and wide-ranging data produced by such analyses?
3. How can we (semi)-automatically compose these analyses to provide more complex higher-level analyses?

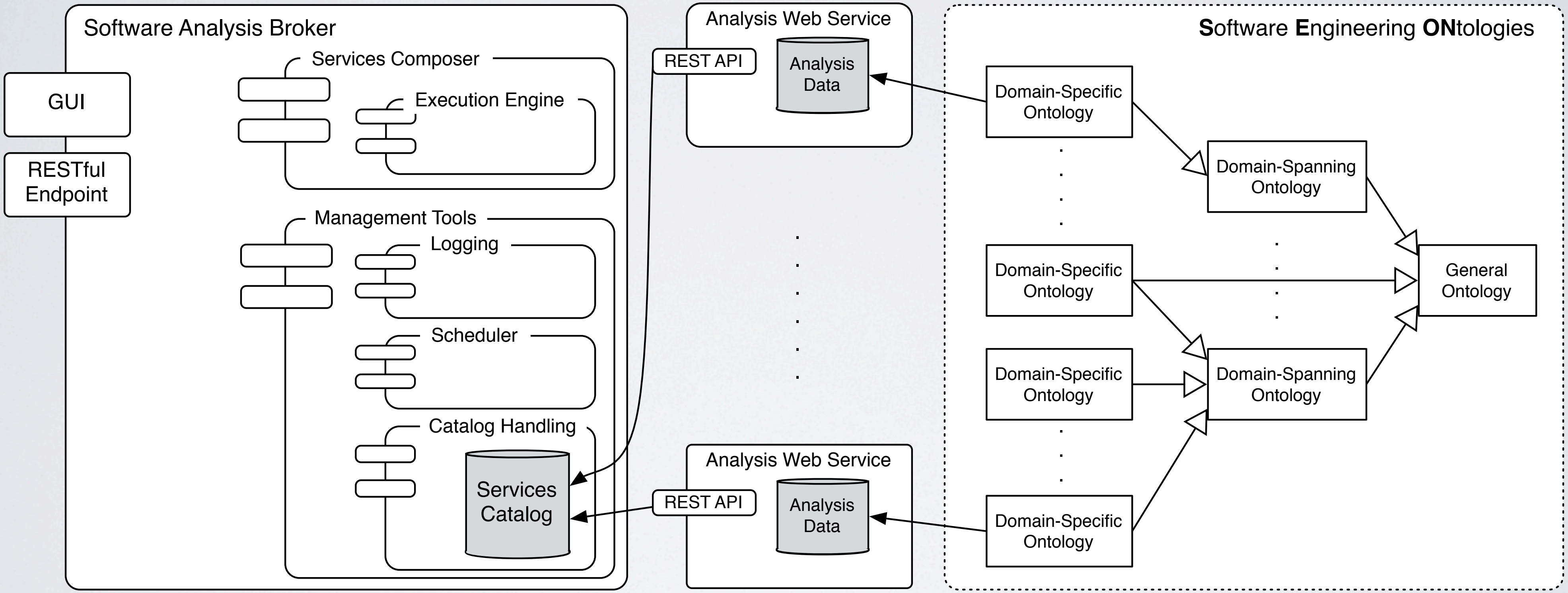
THE NEED FOR A DIFFERENT APPROACH

1. How can we offer the functionalities of this plethora of evolution analyses in a consistent and efficient way?
2. How can we effectively describe the heterogeneous and wide-ranging data produced by such analyses?
3. How can we (semi)-automatically compose these analyses to provide more complex higher-level analyses?
4. **What is the impact of this new approach?**

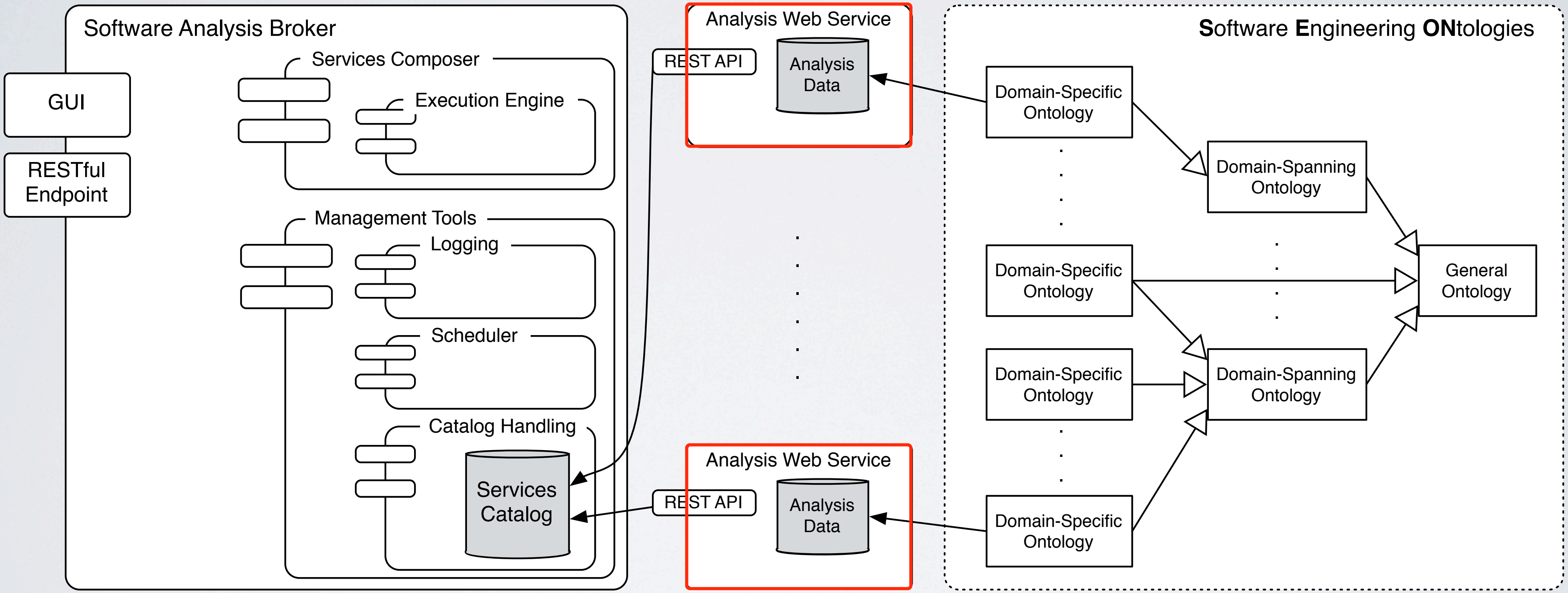
ENTER SOFAS



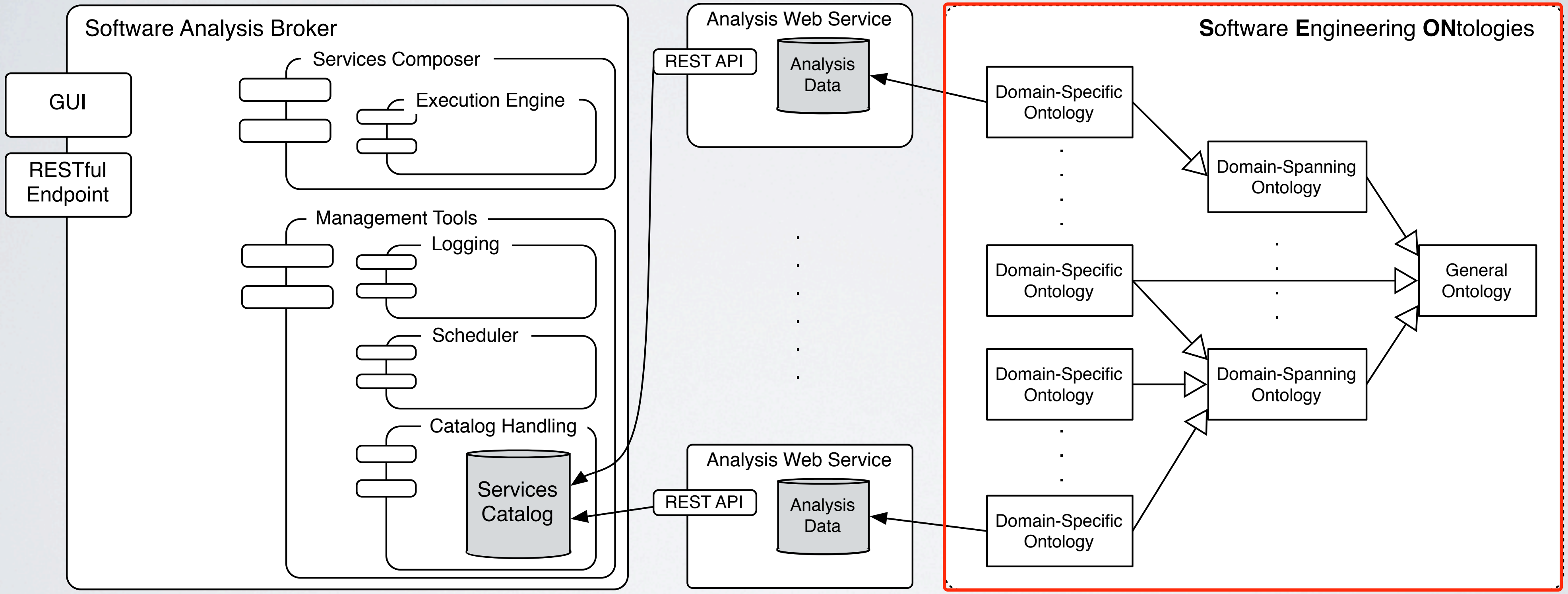
ENTER SOFAS


Software Analysis Broker
GUI
RESTful Endpoint
Services Composer
Execution Engine
Management Tools
Logging
Scheduler
Catalog Handling
Services Catalog
Analysis Web Service
REST API
Analysis Data
Analysis Web Service
REST API
Analysis Data
Software Engineering ONtologies
Domain-Specific Ontology
Domain-Spanning Ontology
Domain-Specific Ontology
General Ontology
Domain-Specific Ontology
Domain-Spanning Ontology
Domain-Specific Ontology

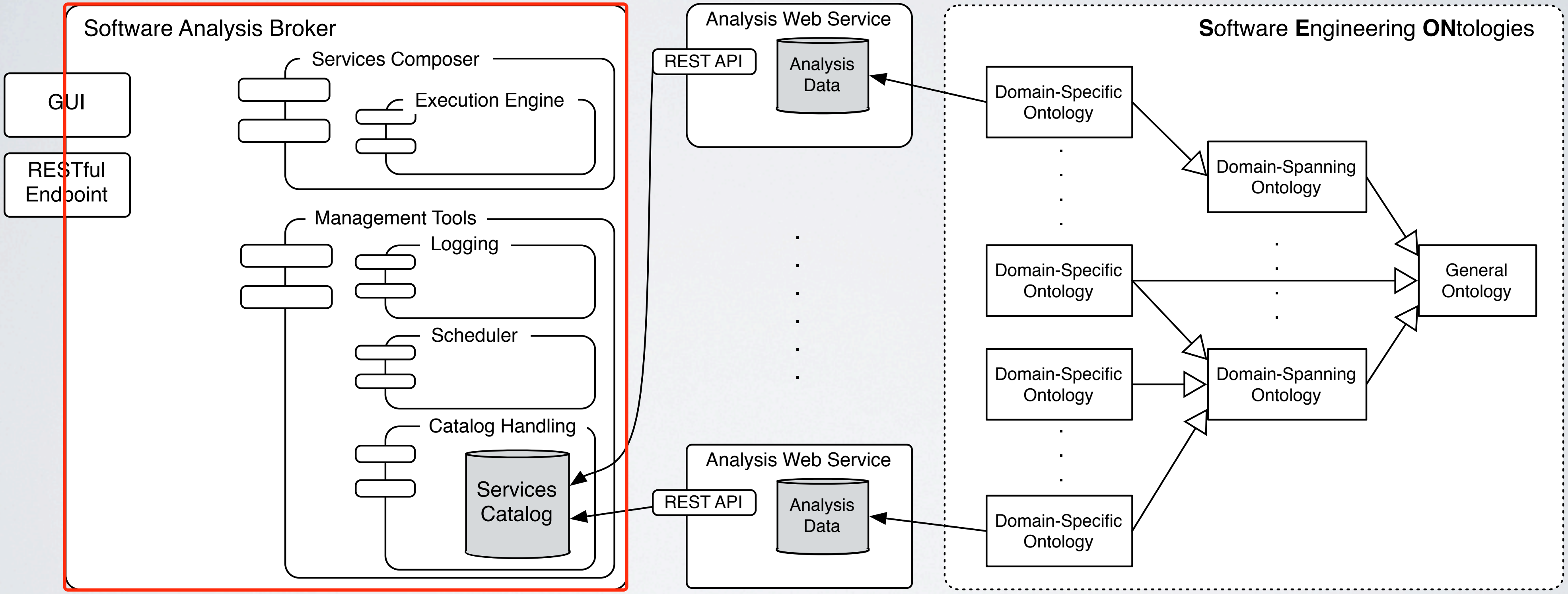
ENTER SOFAS

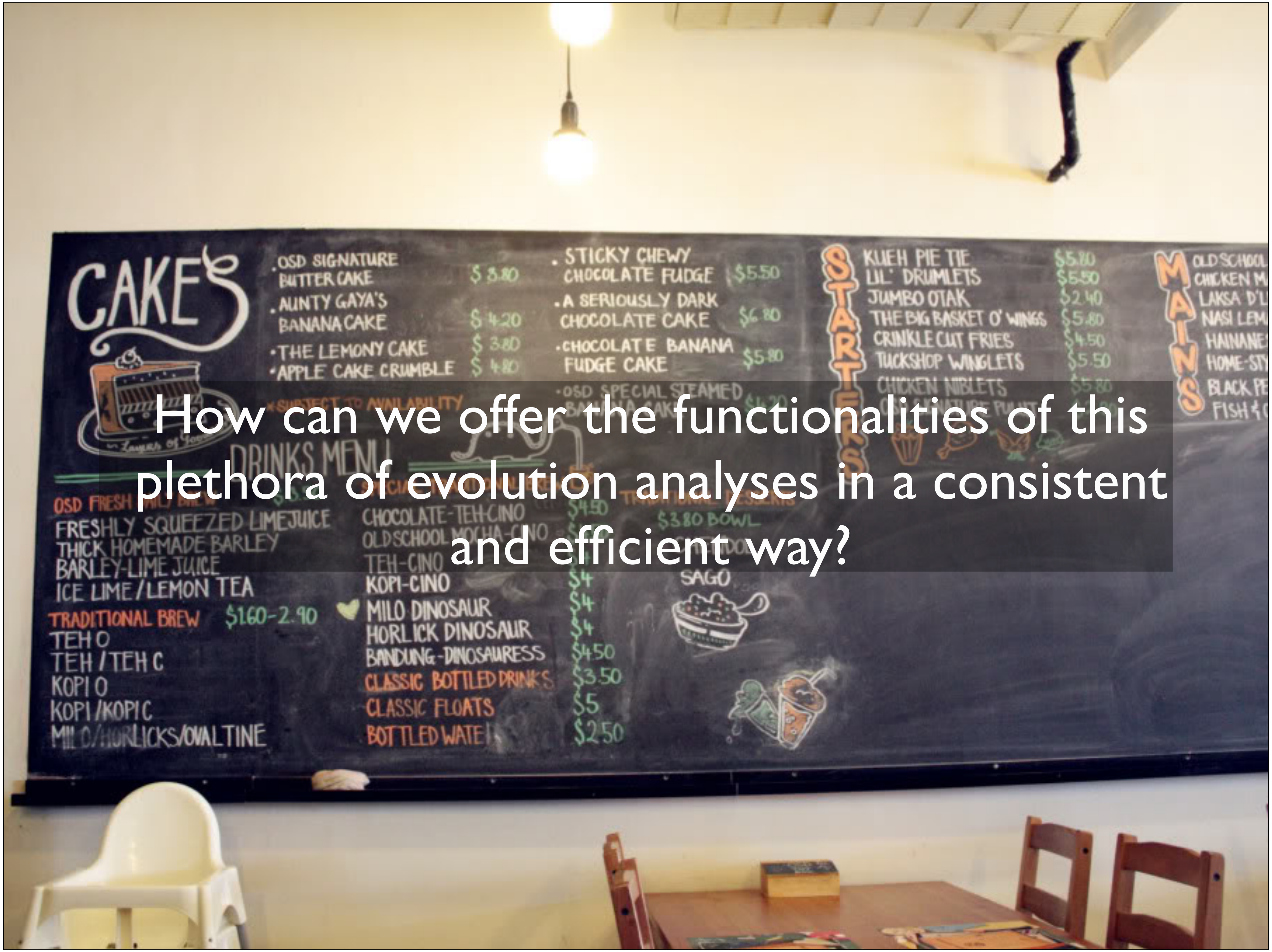

Software Analysis Broker
GUI
RESTful Endpoint
Services Composer
Execution Engine
Management Tools
Logging
Scheduler
Catalog Handling
Services Catalog
Analysis Web Service
REST API
Analysis Data
Analysis Web Service
REST API
Analysis Data
Software Engineering ONtologies
Domain-Specific Ontology
Domain-Spanning Ontology
Domain-Specific Ontology
General Ontology
Domain-Specific Ontology
Domain-Spanning Ontology
Domain-Specific Ontology

ENTER SOFAS


Software Analysis Broker
GUI
RESTful Endpoint
Services Composer
Execution Engine
Management Tools
Logging
Scheduler
Catalog Handling
Services Catalog
Analysis Web Service
REST API
Analysis Data
Analysis Web Service
REST API
Analysis Data
Software Engineering ONtologies
Domain-Specific Ontology
Domain-Spanning Ontology
Domain-Specific Ontology
General Ontology
Domain-Specific Ontology
Domain-Spanning Ontology
Domain-Specific Ontology

ENTER SOFAS


GUI
RESTful Endpoint
Software Analysis Broker
Services Composer
Execution Engine
Management Tools
Logging
Scheduler
Catalog Handling
Services Catalog
Analysis Web Service
REST API
Analysis Data
Analysis Web Service
REST API
Analysis Data
Software Engineering ONtologies
Domain-Specific Ontology
Domain-Spanning Ontology
Domain-Specific Ontology
General Ontology
Domain-Specific Ontology
Domain-Spanning Ontology
Domain-Specific Ontology



How can we offer the functionalities of this plethora of evolution analyses in a consistent and efficient way?

SOFTWARE ANALYSIS AS A SERVICE

SOFTWARE ANALYSIS AS A SERVICE

PLEASE, MAKE ME A CHOCOLATE CAKE
CHOCO

ANALYSIS WEB SERVICES

- Analysis service (habanero.ifi.uzh.ch/svnImporter/analyses)
  - **GET** - Lists al the existing analyses
  - **POST** - Creates and runs a new analysis
- Individual analysis (habanero.ifi.uzh.ch/svnImporter/analyses/analysis_1)
  - **GET** - Returns the complete analysis data or the results of a SPARQL query encoded in the analysis URI (SPARQL Endpoint)
  - **HEAD** - Checks if the analysis exists and is available
  - **PUT** - Replaces the analysis
  - **DELETE** - Deletes the analysis

ANALYSIS WEB SERVICES

- More than 30 services:
  - Data gatherers
    - **version control history** extractors
    - **issue tracking history** extractors
  - Basic analyses
    - **metrics** calculators
    - **source code model** extractors
    - **code clones** calculator
    - **change types** distiller
  - Composite analyses
    - **issue-revision** linkers
    - **code disharmonies** detector
    - **defect** predictors

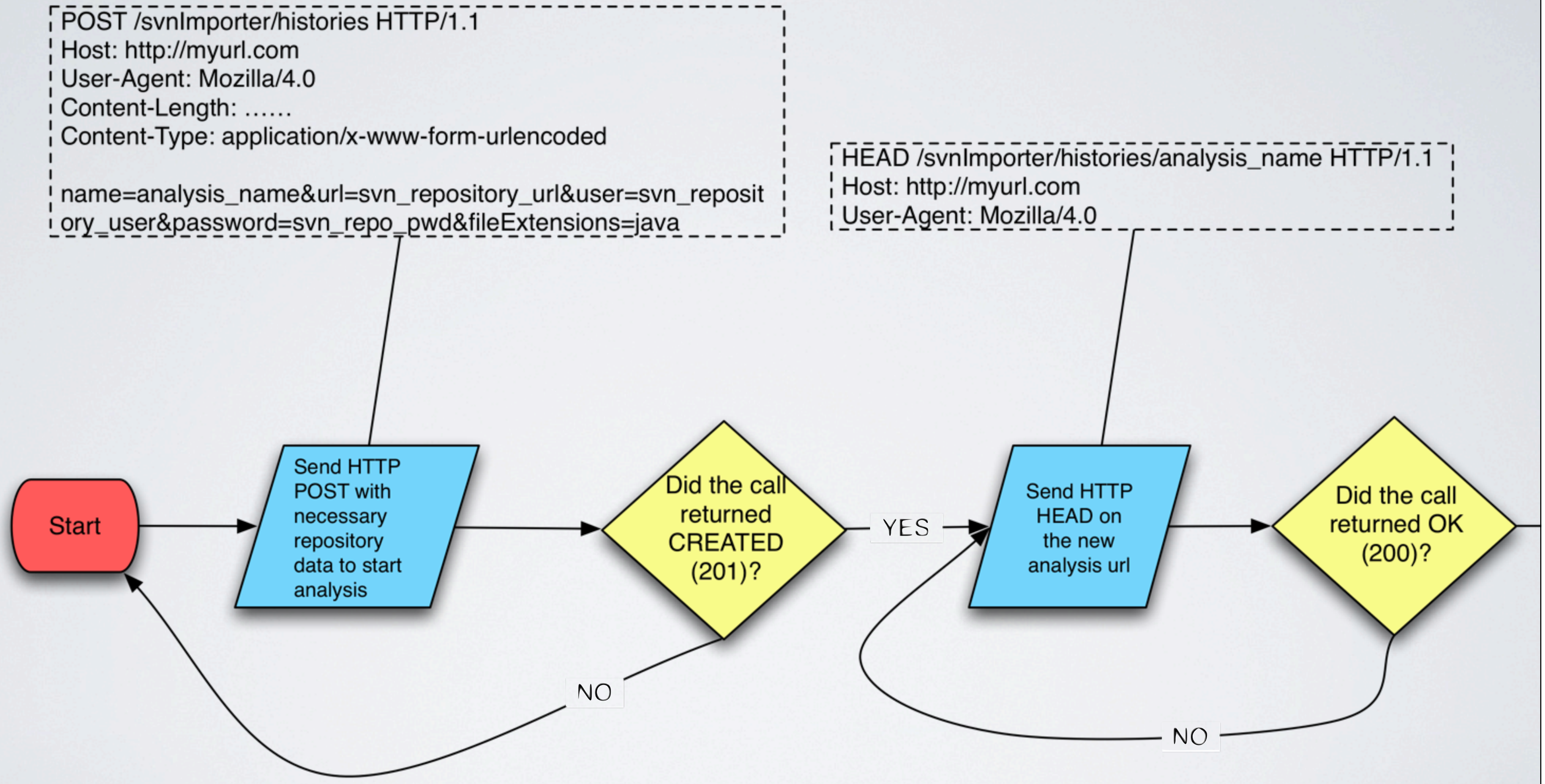
HOW TO USE SOFAS' RESTFUL SERVICES (1)

- Starting an analysis
  - access via HTTP, cURL, or Java/Jersey
  - **send a post** request to habanero.ifi.uzh.ch/svnImporter/histories together with the data needed in XML format or in an HTML form
  - **returns the URL** of the newly created analyses (through which its data can then be fetched)
  - the actual data will be available only when the entire analysis completed
  - the service will send an email when the analysis is completed (however, a program can periodically send an HTTP HEAD request on the analysis URL)

HOW TO USE SOFAS' RESTFUL SERVICES (2)

- Fetching the data
  - Fetching the data is all about URLs and SPARQL
  - All the **data produced**, since its based on RDFs ontologies, has a unique **URI**
  - Results come either in **plain text** (more for 'human consumption') or as **SPARQL** results
- Further querying the data
  - For that all the services also offer a SPARQL **endpoint** for each existing analysis.
  - SPARQL **queries** can be directly sent to the analyses, returning the results as SPARQL Results XML

SOFAS FLOW (1)



```
POST /svnImporter/histories HTTP/1.1
Host: http://myurl.com
User-Agent: Mozilla/4.0
Content-Length: ......
Content-Type: application/x-www-form-urlencoded

name=analysis_name&url=svn_repository_url&user=svn_reposit
ory_user&password=svn_repo_pwd&fileExtensions=java
```

```
HEAD /svnImporter/histories/analysis_name HTTP/1.1
Host: http://myurl.com
User-Agent: Mozilla/4.0
```

Start

Send HTTP POST with necessary repository data to start analysis

Did the call returned CREATED (201)?

YES

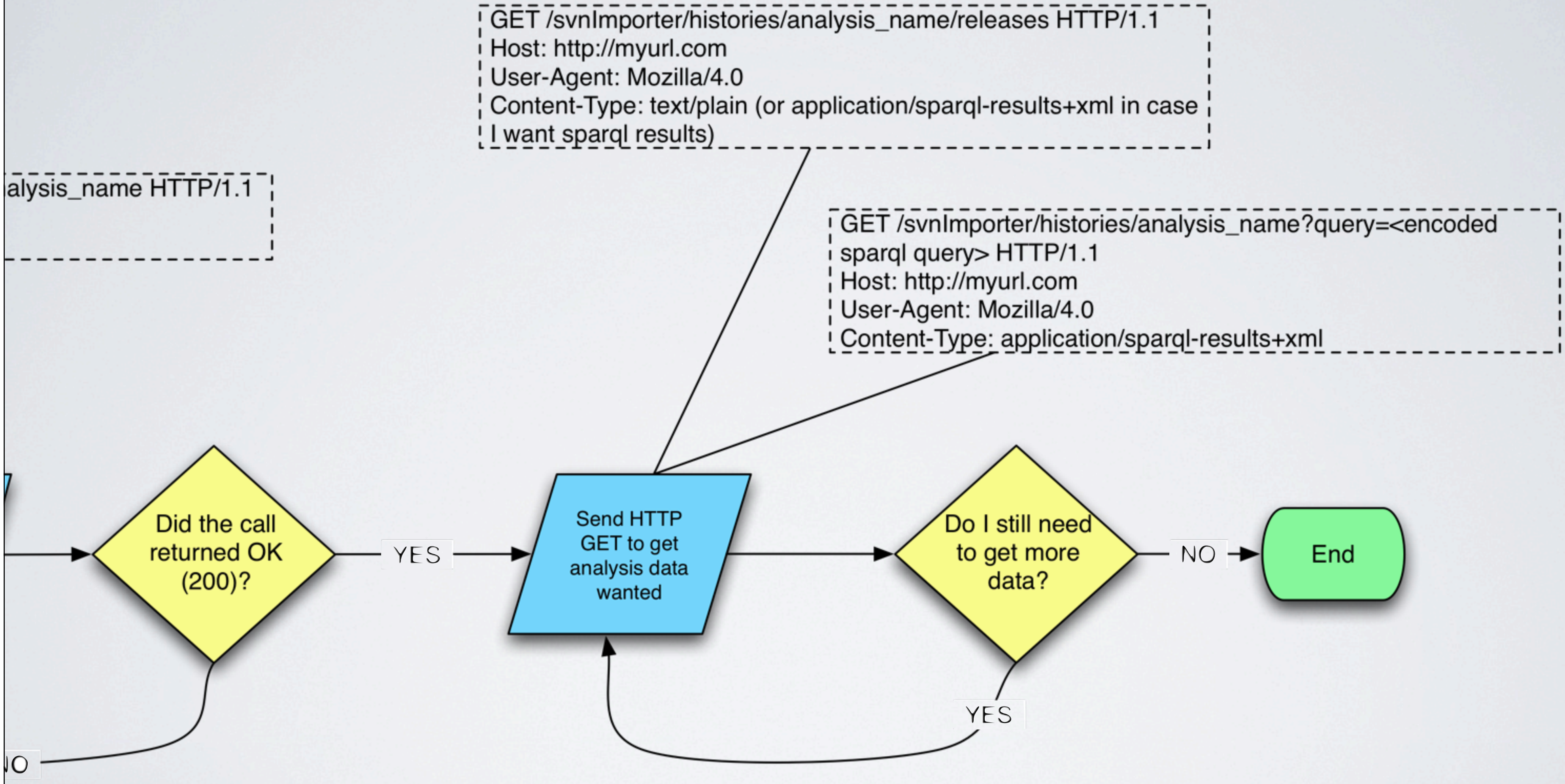
NO

Send HTTP HEAD on the new analysis url

Did the call returned OK (200)?

NO

SOFAS FLOW (2)



```
GET /svnImporter/histories/analysis_name/releases HTTP/1.1
Host: http://myurl.com
User-Agent: Mozilla/4.0
Content-Type: text/plain (or application/sparql-results+xml in case
I want sparql results)
```

```
alysis_name HTTP/1.1
```

```
GET /svnImporter/histories/analysis_name?query=<encoded
sparql query> HTTP/1.1
Host: http://myurl.com
User-Agent: Mozilla/4.0
Content-Type: application/sparql-results+xml
```

Did the call returned OK (200)?

YES

Send HTTP GET to get analysis data wanted

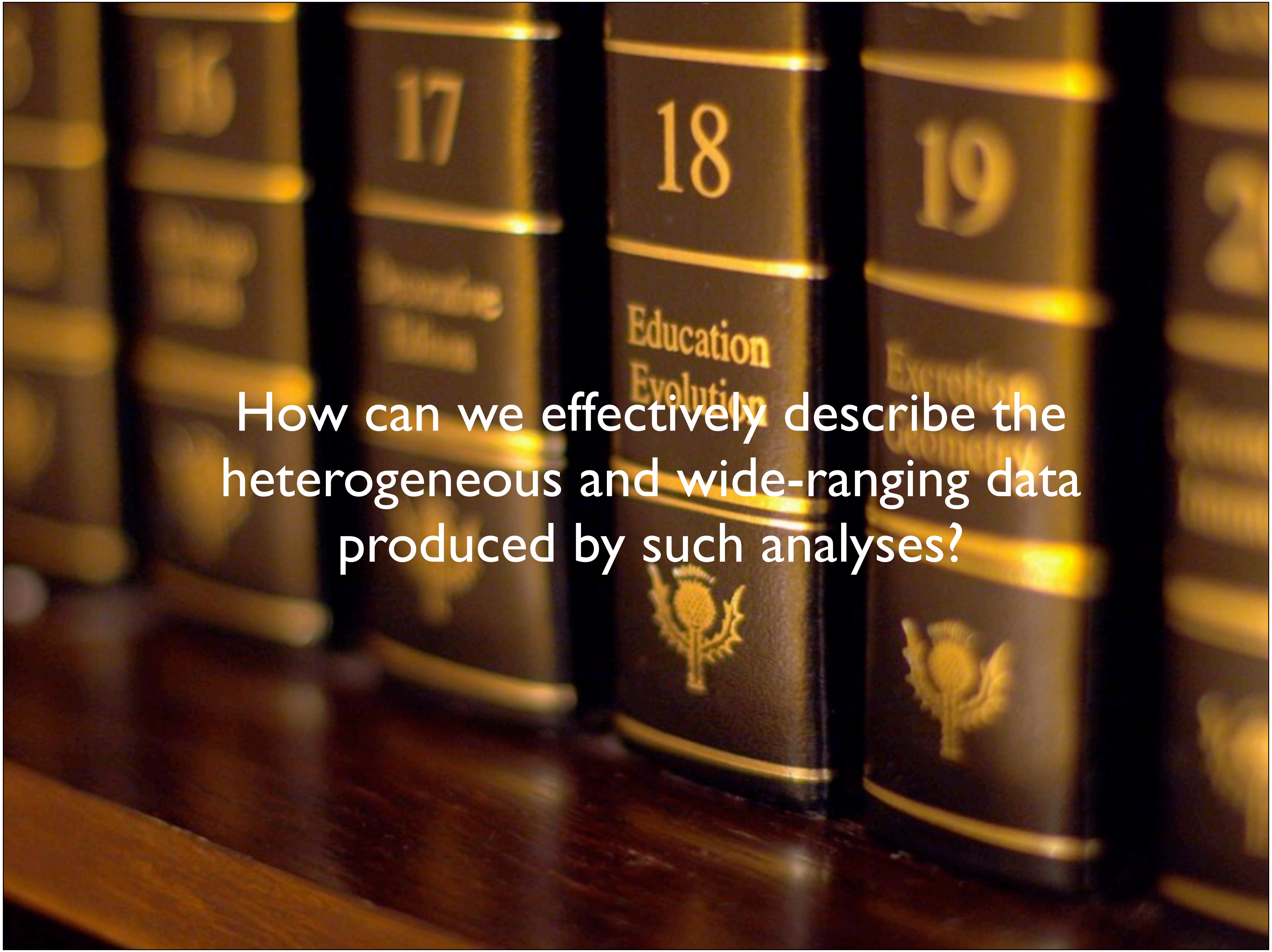
Do I still need to get more data?

NO

End

YES

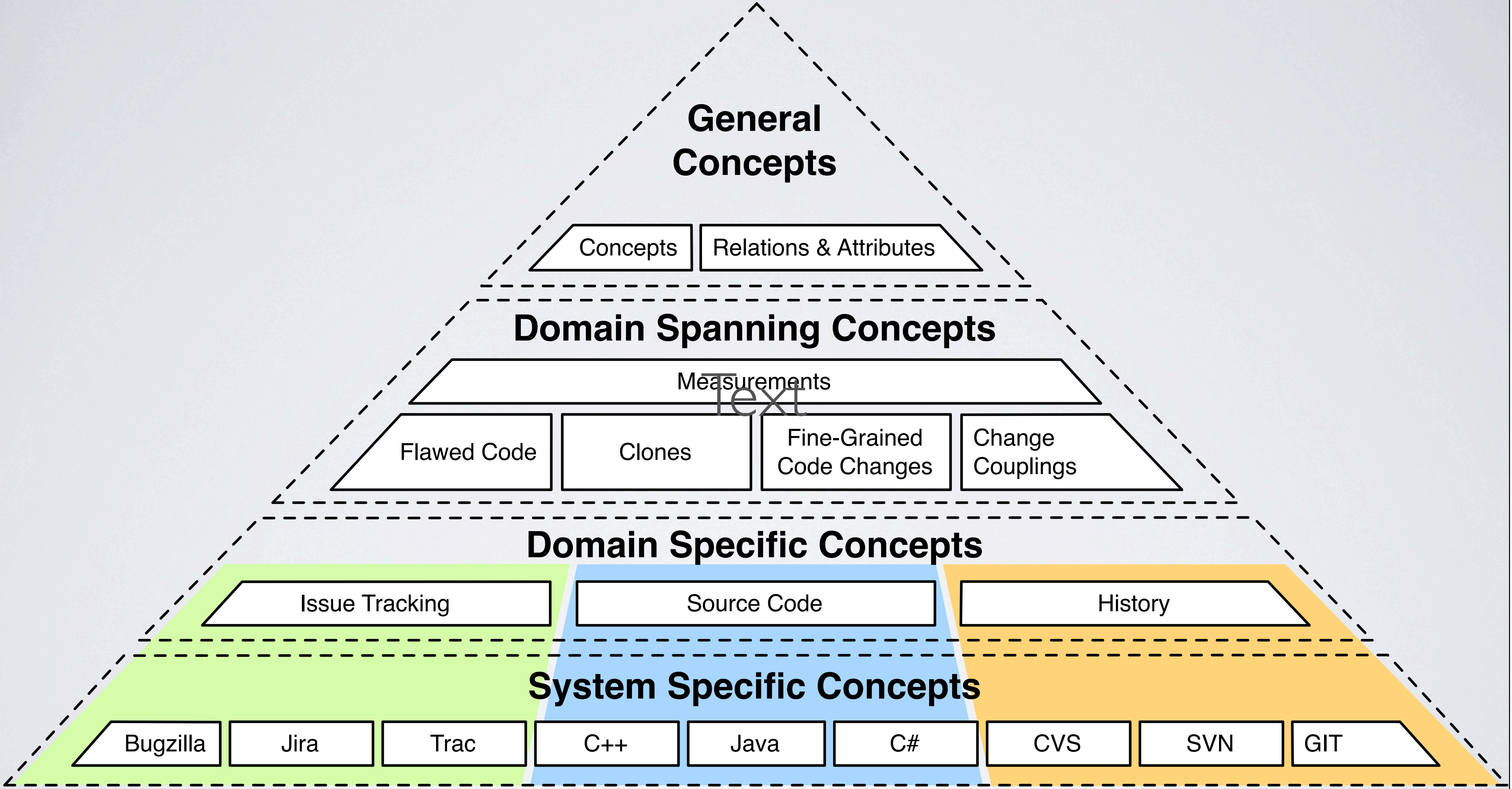
O



How can we effectively describe the heterogeneous and wide-ranging data produced by such analyses?

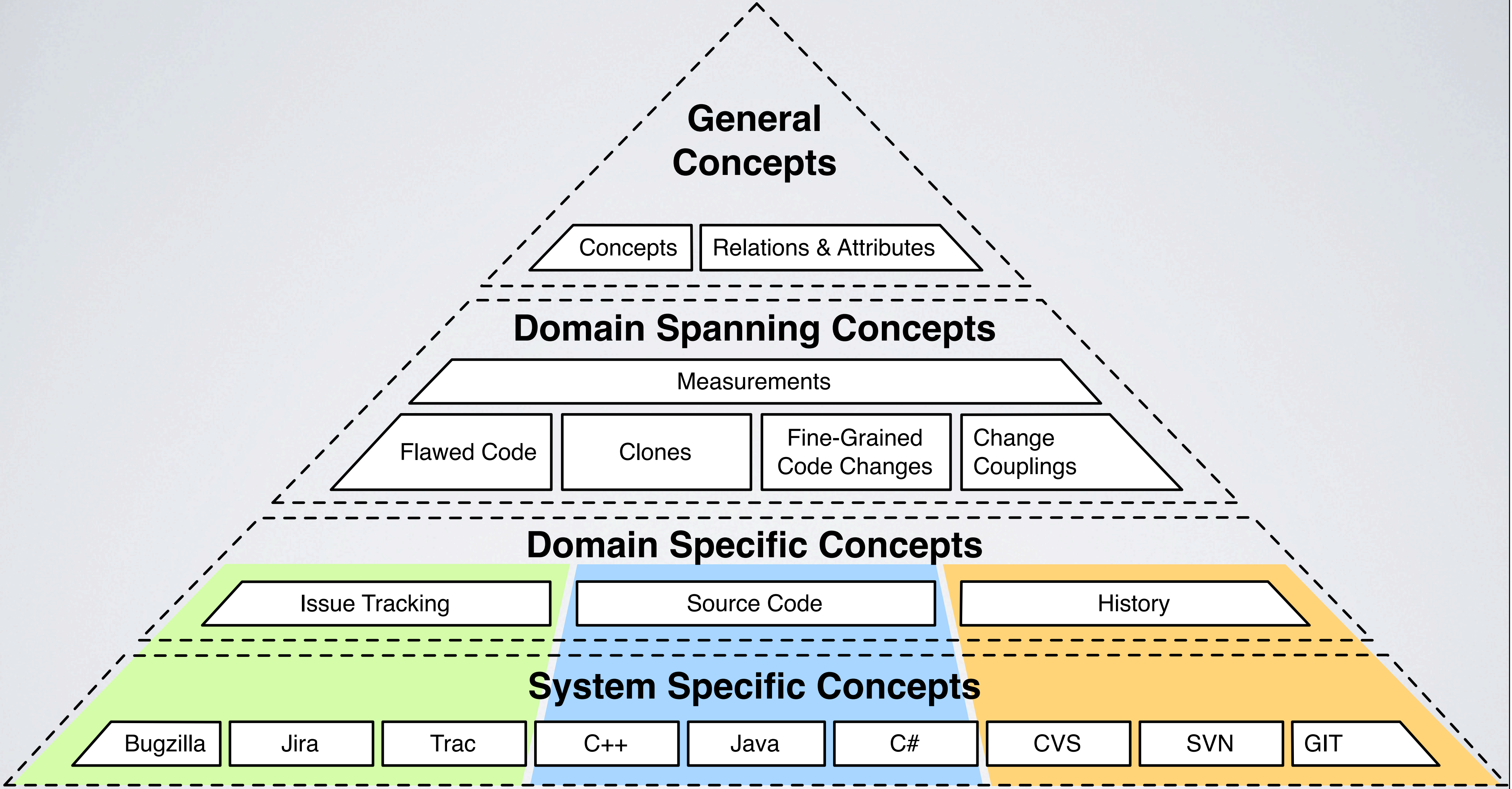
SEON

se-on.org



SEON

se-on.org

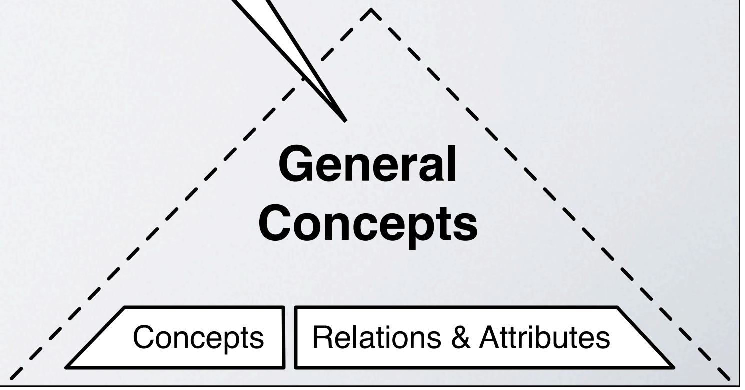


SEON

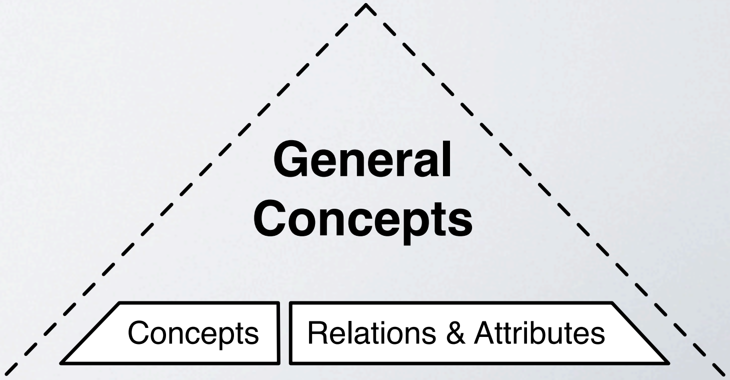
se-on.org

Classes:
Activity, Stakeholder, Artifact, File,...

Properties:
hasSize,createdOn, hasAuthor, dependsOn...

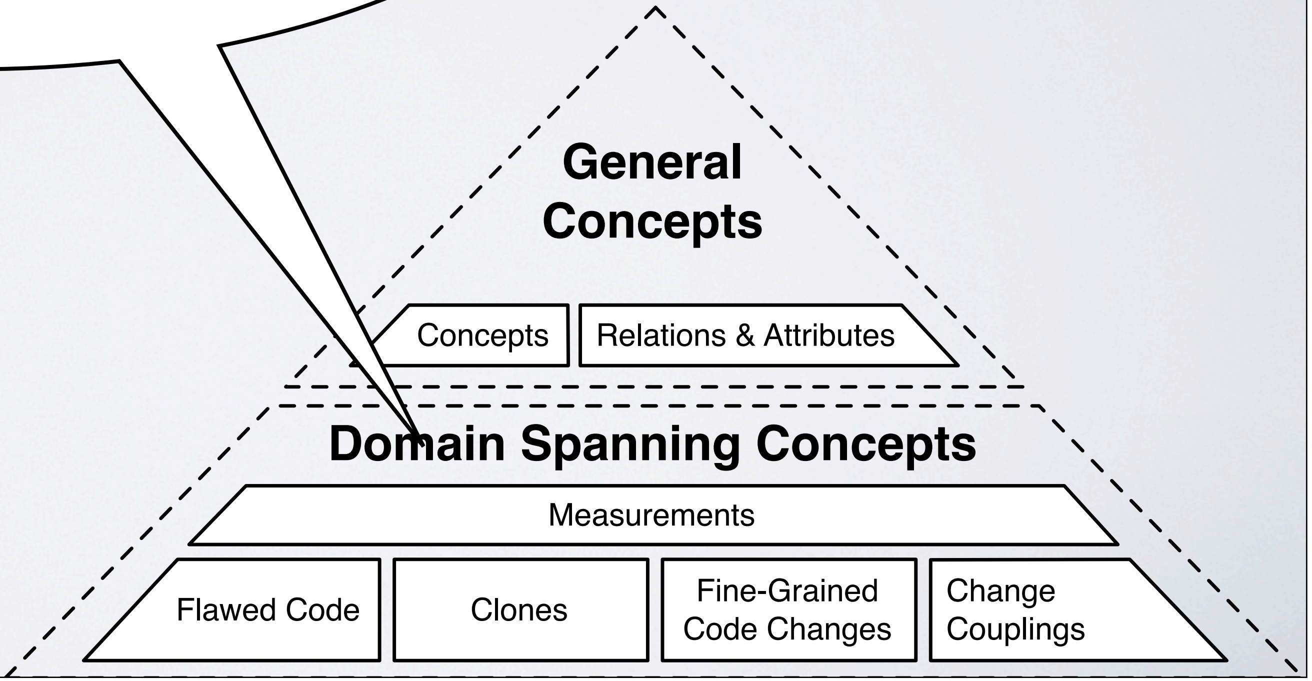


SEON



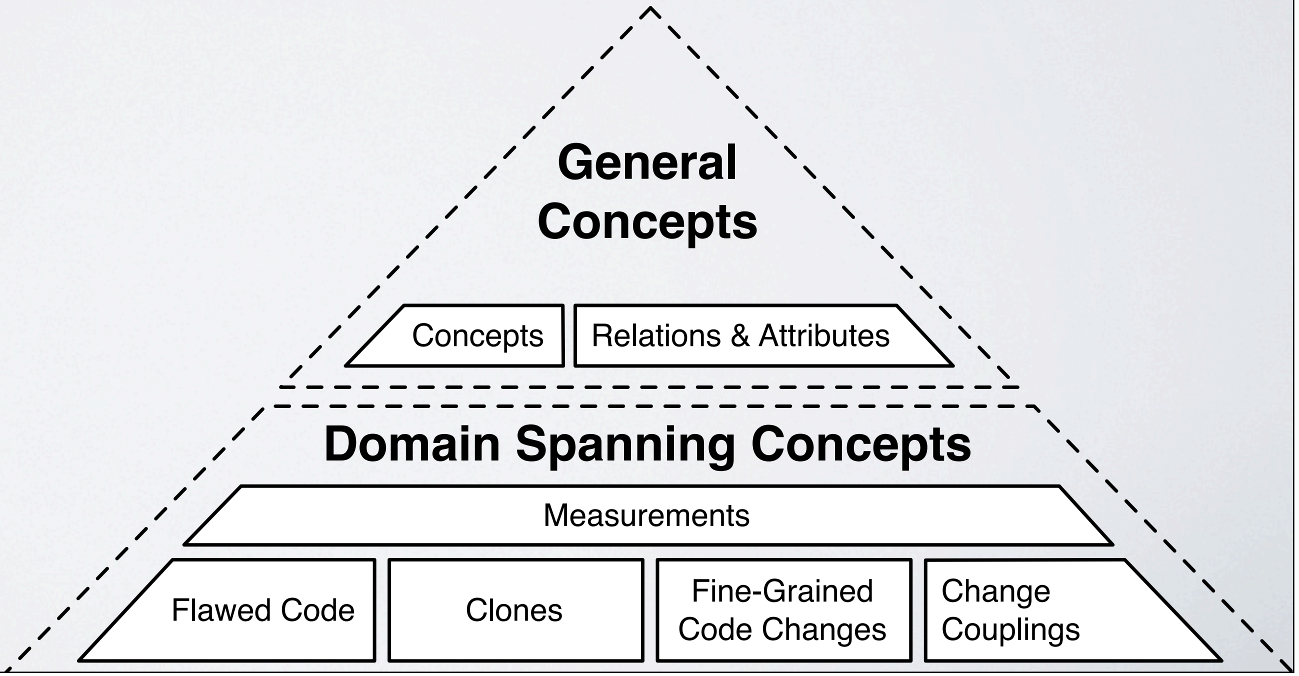
SEON

Classes:
Measure, TypeOfCoupling, Clone,...
Properties:
fixesIssue,isFixedIn, isCoupledWith...



SEON

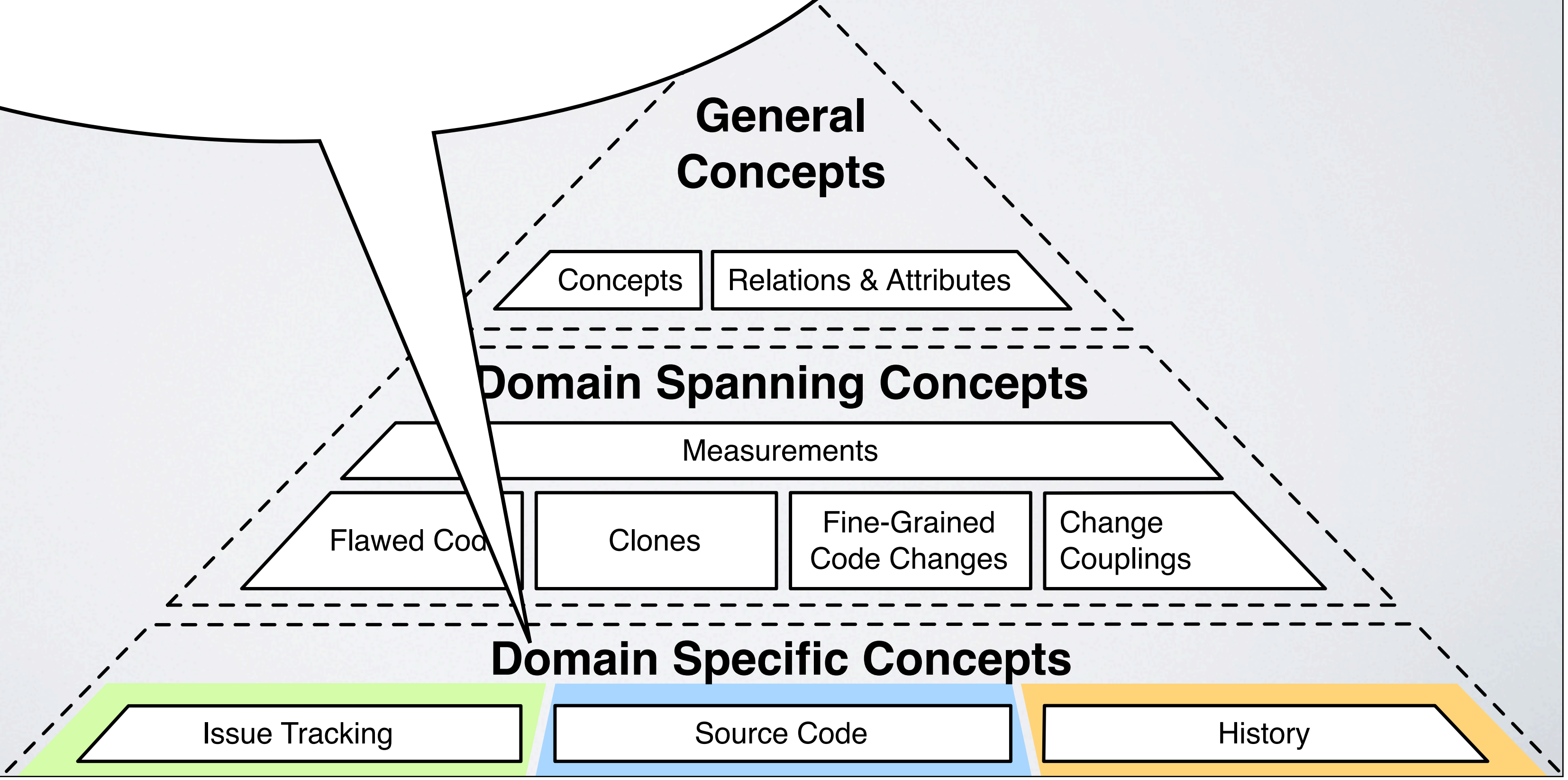
se-on.org



SEON

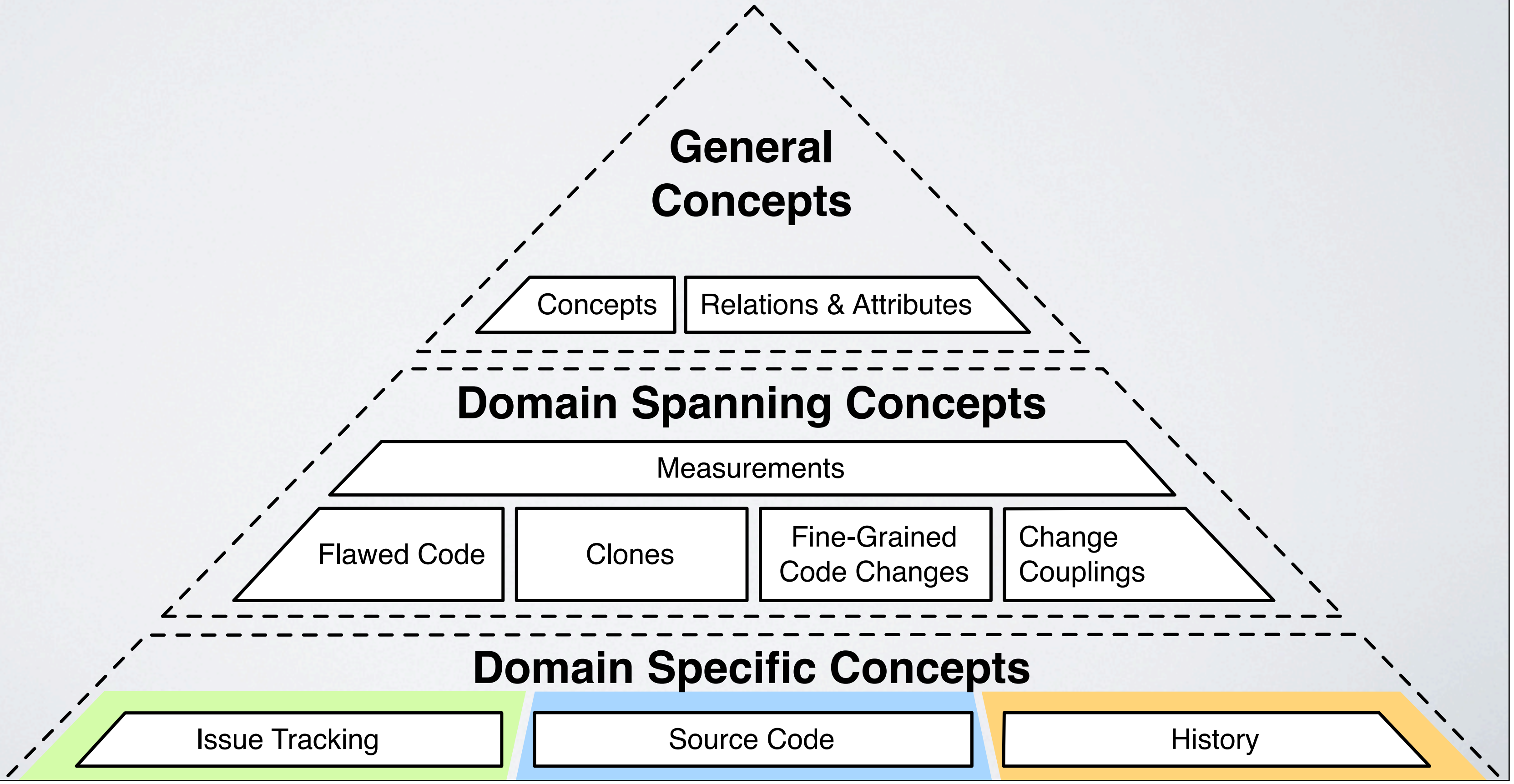
se-on.org

Classes:
Issue, Release, Class, Method...
Properties:
commits,reports, hasMethod...



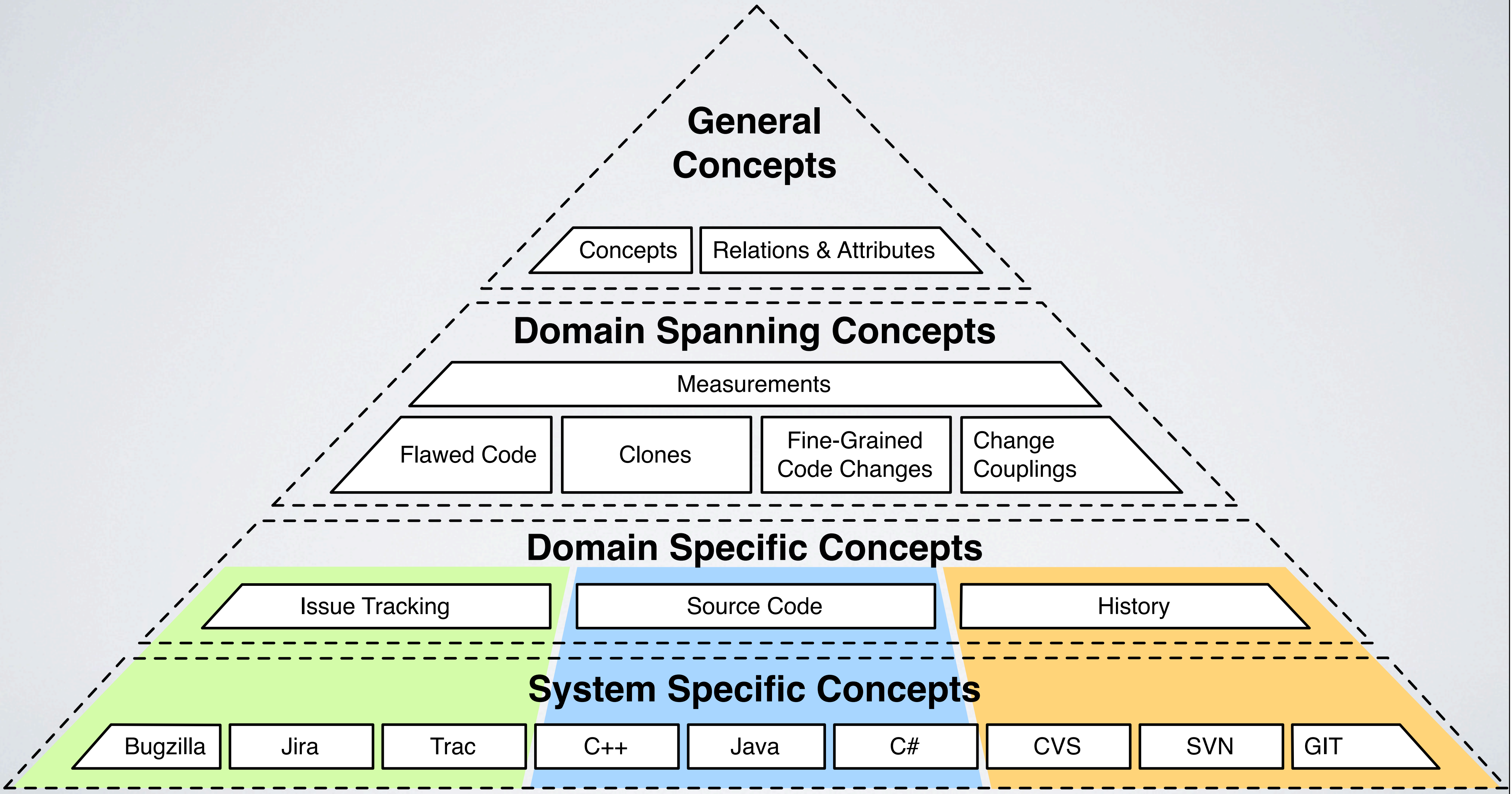
SEON

se-on.org



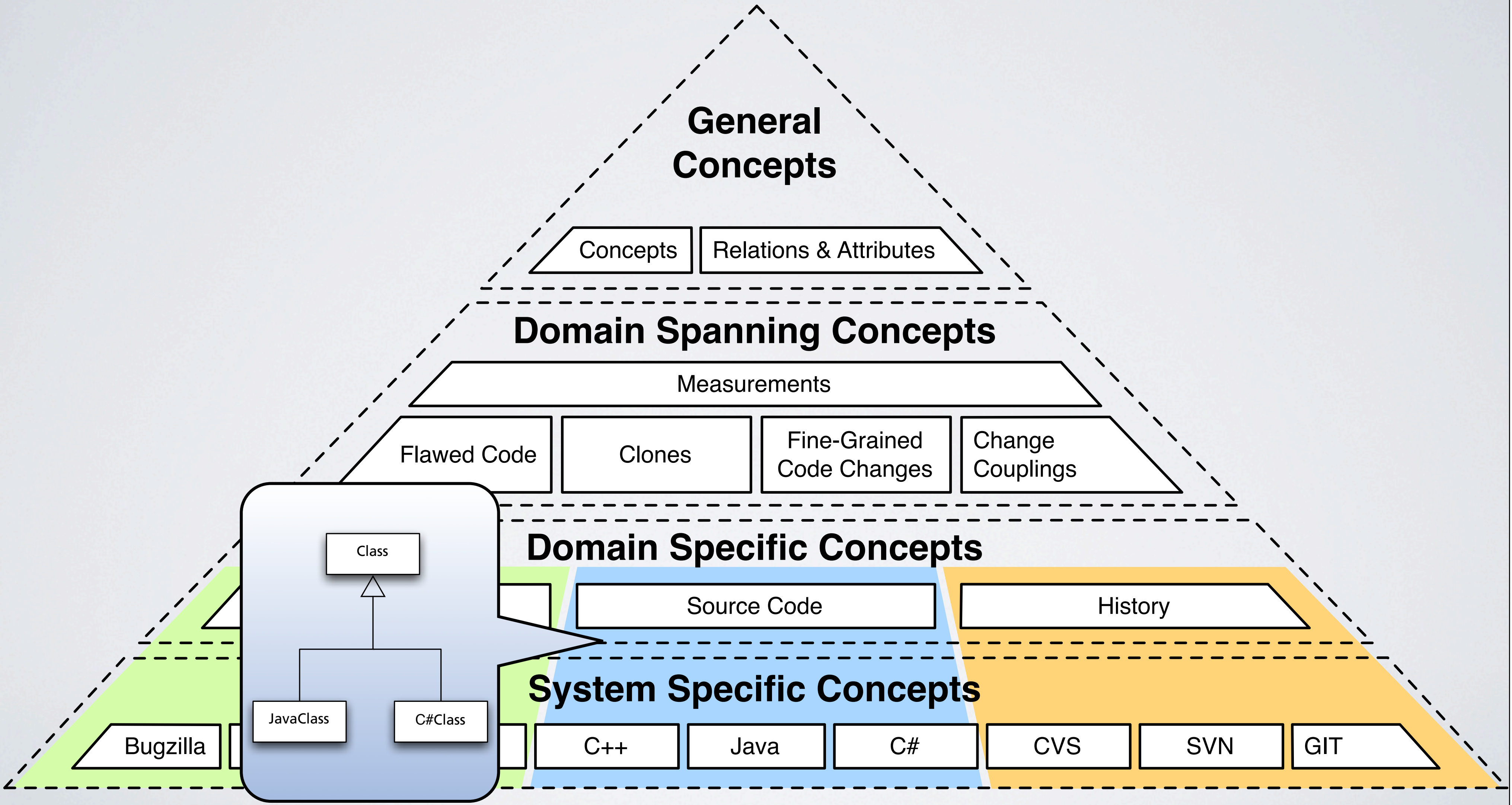
SEON

se-on.org



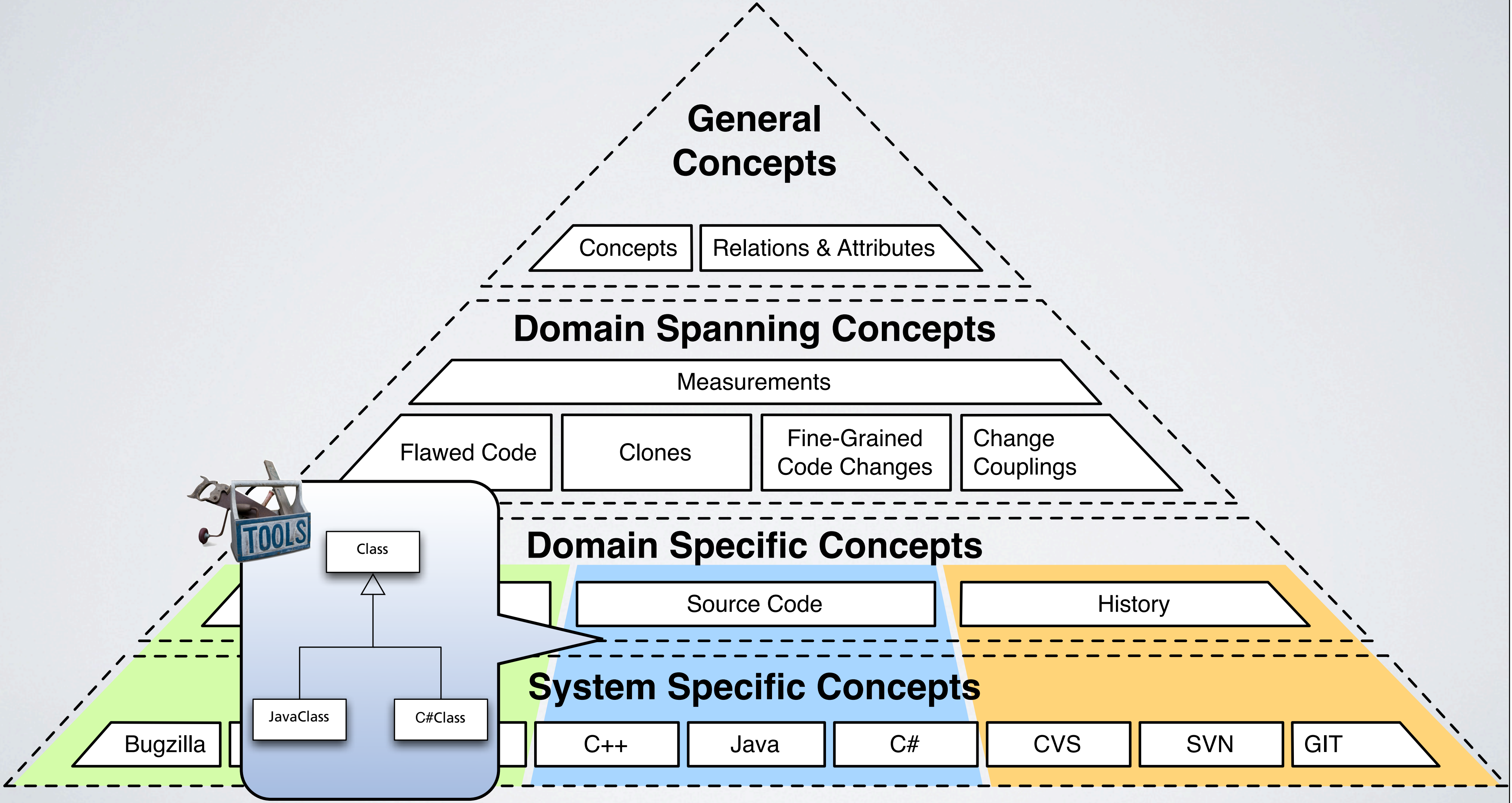
SEON

se-on.org



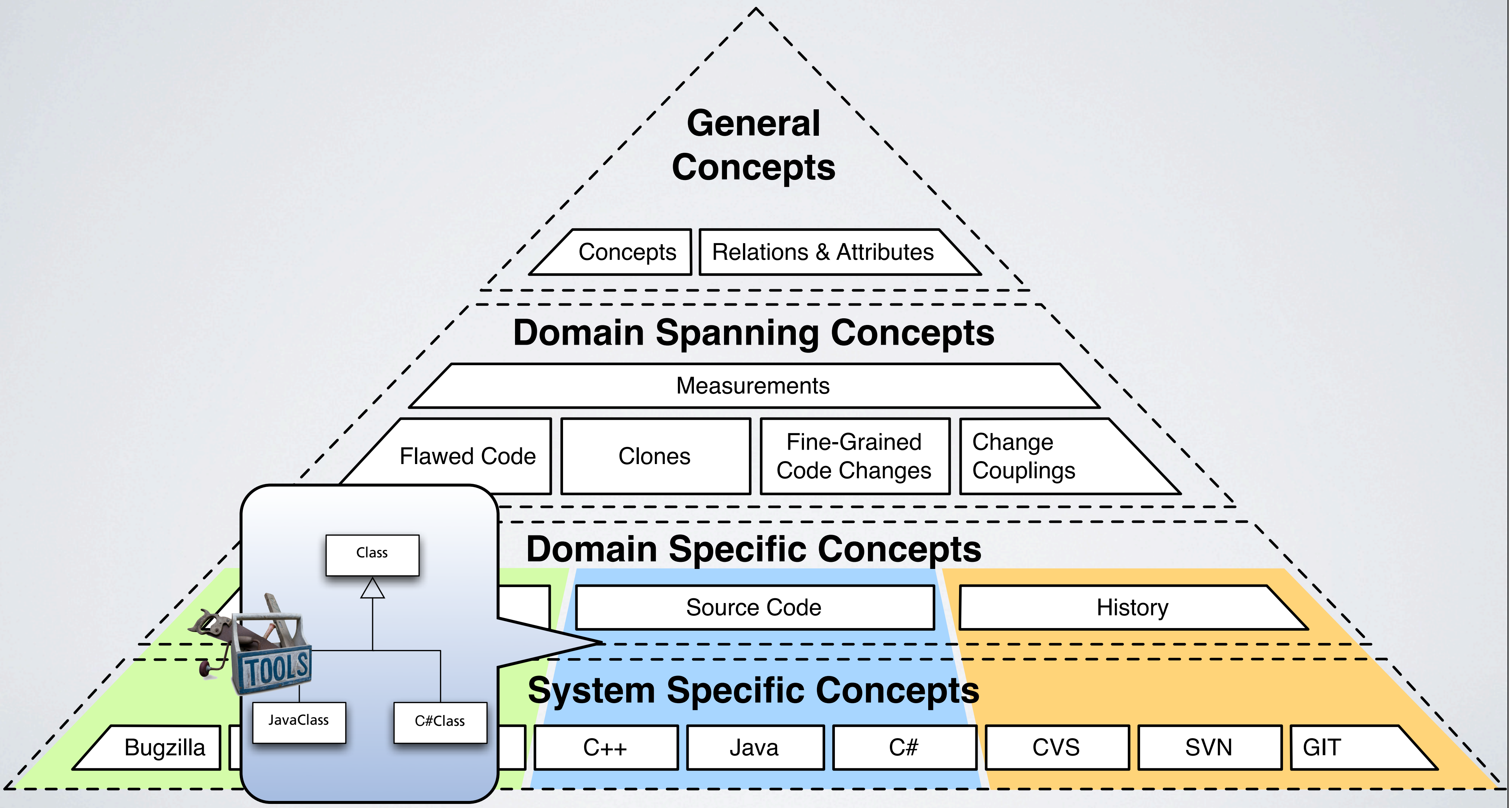
SEON

se-on.org



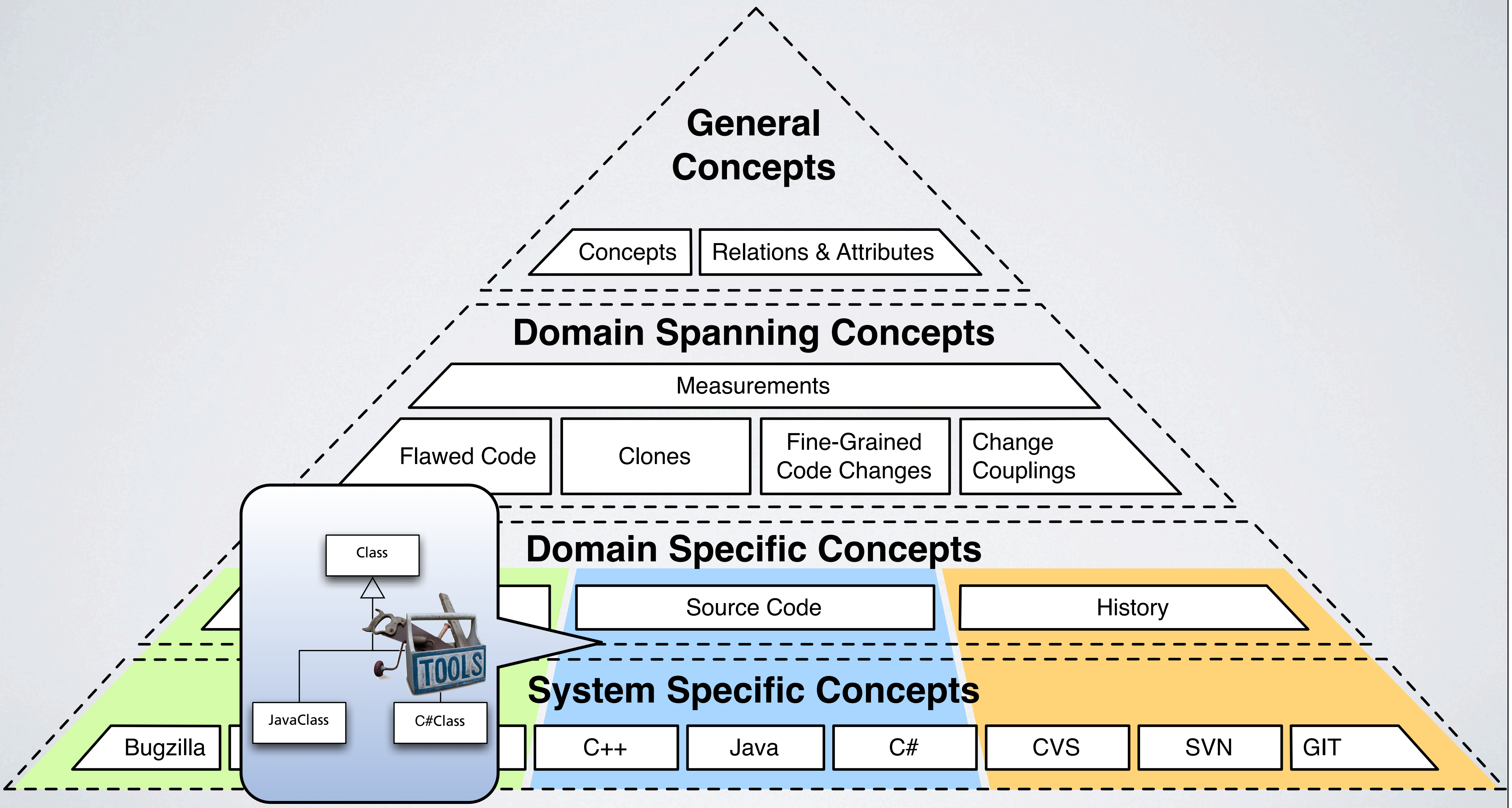
SEON

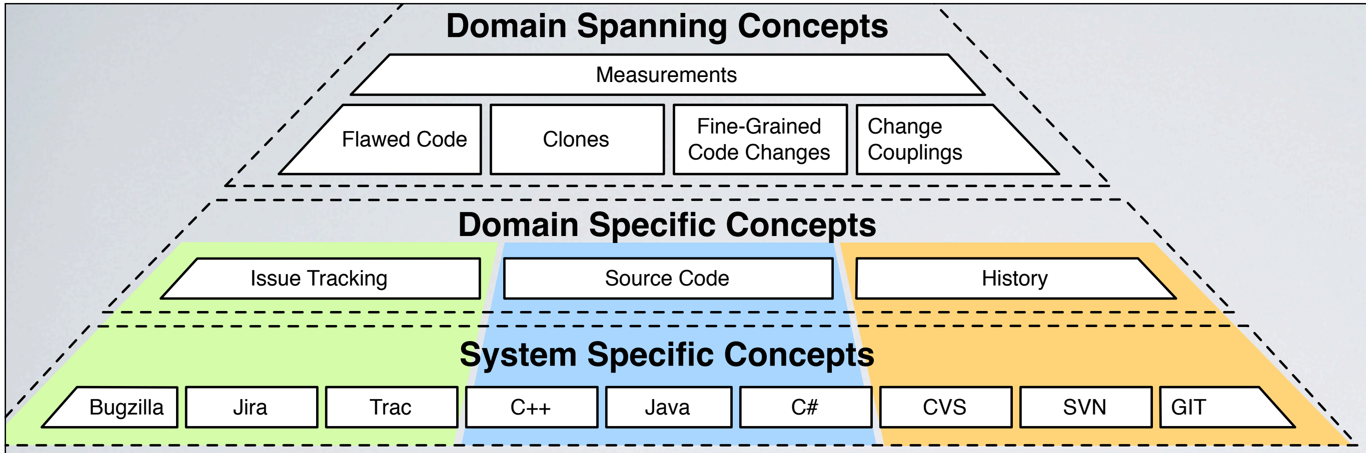
se-on.org



SEON

se-on.org





Domain Spanning Concepts

Measurements

Flawed Code

Clones

Fine-Grained Code Changes

Change Couplings

Domain Specific Concepts

Issue Tracking

Source Code

History

System Specific Concepts

Bugzilla

Jira

Trac

C++

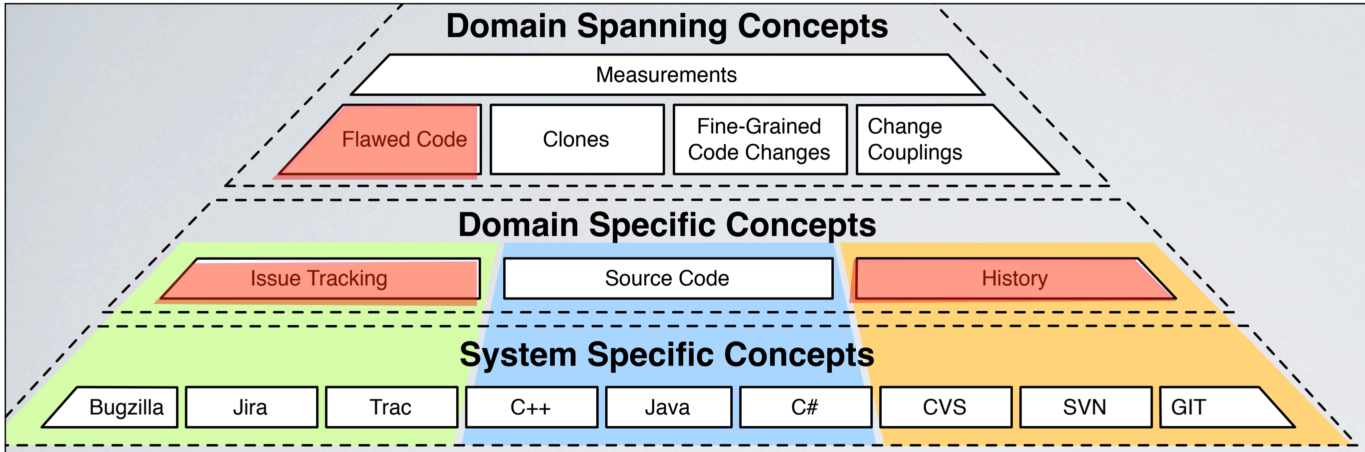
Java

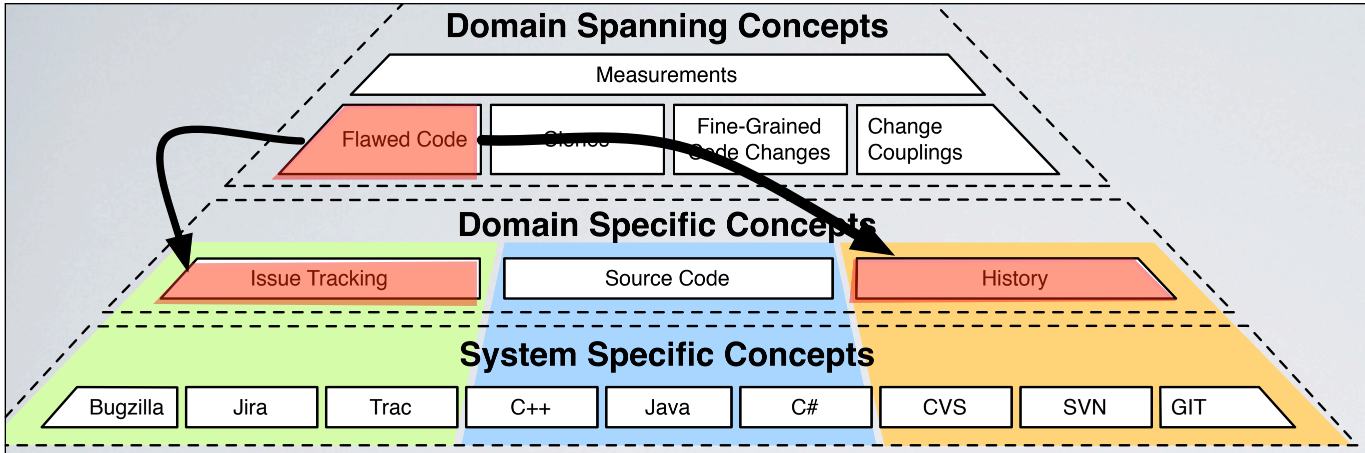
C#

CVS

SVN

GIT


Domain Spanning Concepts
Measurements
Flawed Code
Clones
Fine-Grained Code Changes
Change Couplings
Domain Specific Concepts
Issue Tracking
Source Code
History
System Specific Concepts
Bugzilla
Jira
Trac
C++
Java
C#
CVS
SVN
GIT



Domain Spanning Concepts

Measurements

Flawed Code

Clones

Fine-Grained Code Changes

Change Couplings

Domain Specific Concepts

Issue Tracking

Source Code

History

System Specific Concepts

Bugzilla

Jira

Trac

C++

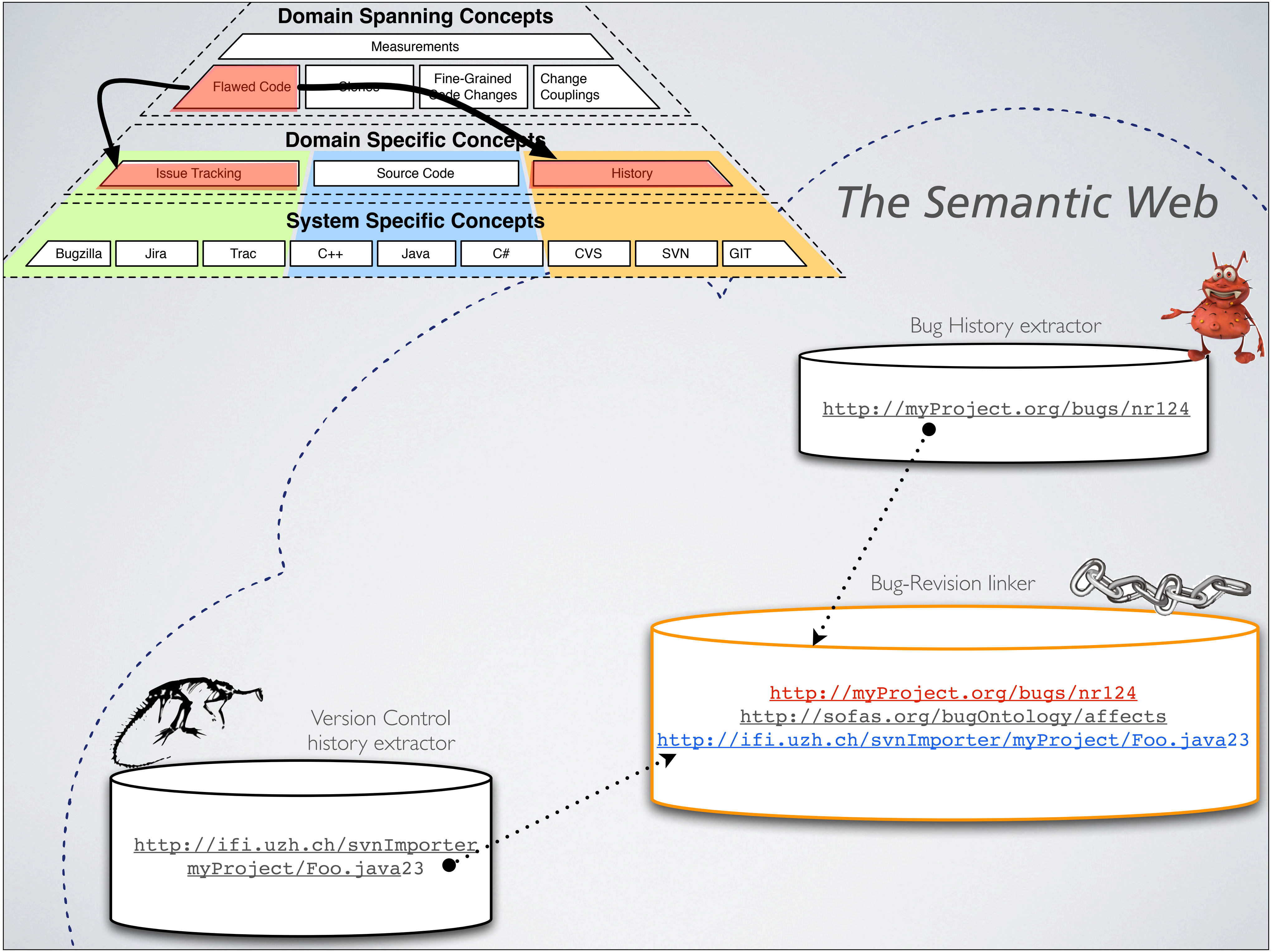
Java

C#

CVS

SVN

GIT



The Semantic Web

Domain Spanning Concepts

Measurements

Flawed Code

Clones

Fine-Grained Code Changes

Change Couplings

Domain Specific Concepts

Issue Tracking

Source Code

History

System Specific Concepts

Bugzilla

Jira

Trac

C++

Java

C#

CVS

SVN

GIT

Bug History extractor

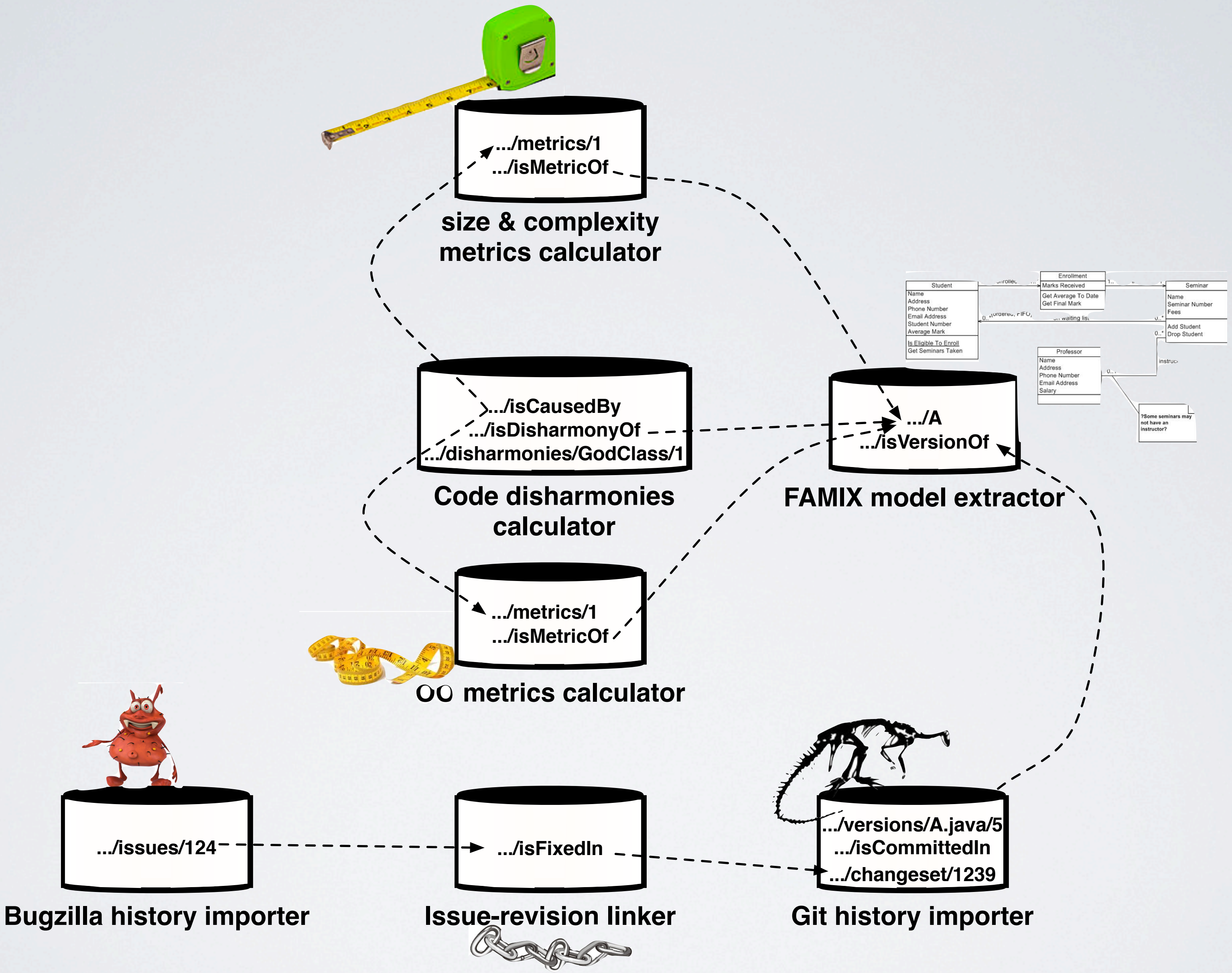
http://myProject.org/bugs/nr124

Bug-Revision linker

http://myProject.org/bugs/nr124
http://sofas.org/bugOntology/affects
http://ifi.uzh.ch/svnImporter/myProject/Foo.java23

Version Control history extractor

http://ifi.uzh.ch/svnImporter
myProject/Foo.java23



.../metrics/1
.../isMetricOf

size & complexity metrics calculator

.../isCausedBy
.../isDisharmonyOf
.../disharmonies/GodClass/1

Code disharmonies calculator

.../A
.../isVersionOf

FAMIX model extractor

.../metrics/1
.../isMetricOf

OO metrics calculator

.../issues/124

Bugzilla history importer

.../isFixedIn

Issue-revision linker

.../versions/A.java/5
.../isCommittedIn
.../changeset/1239

Git history importer

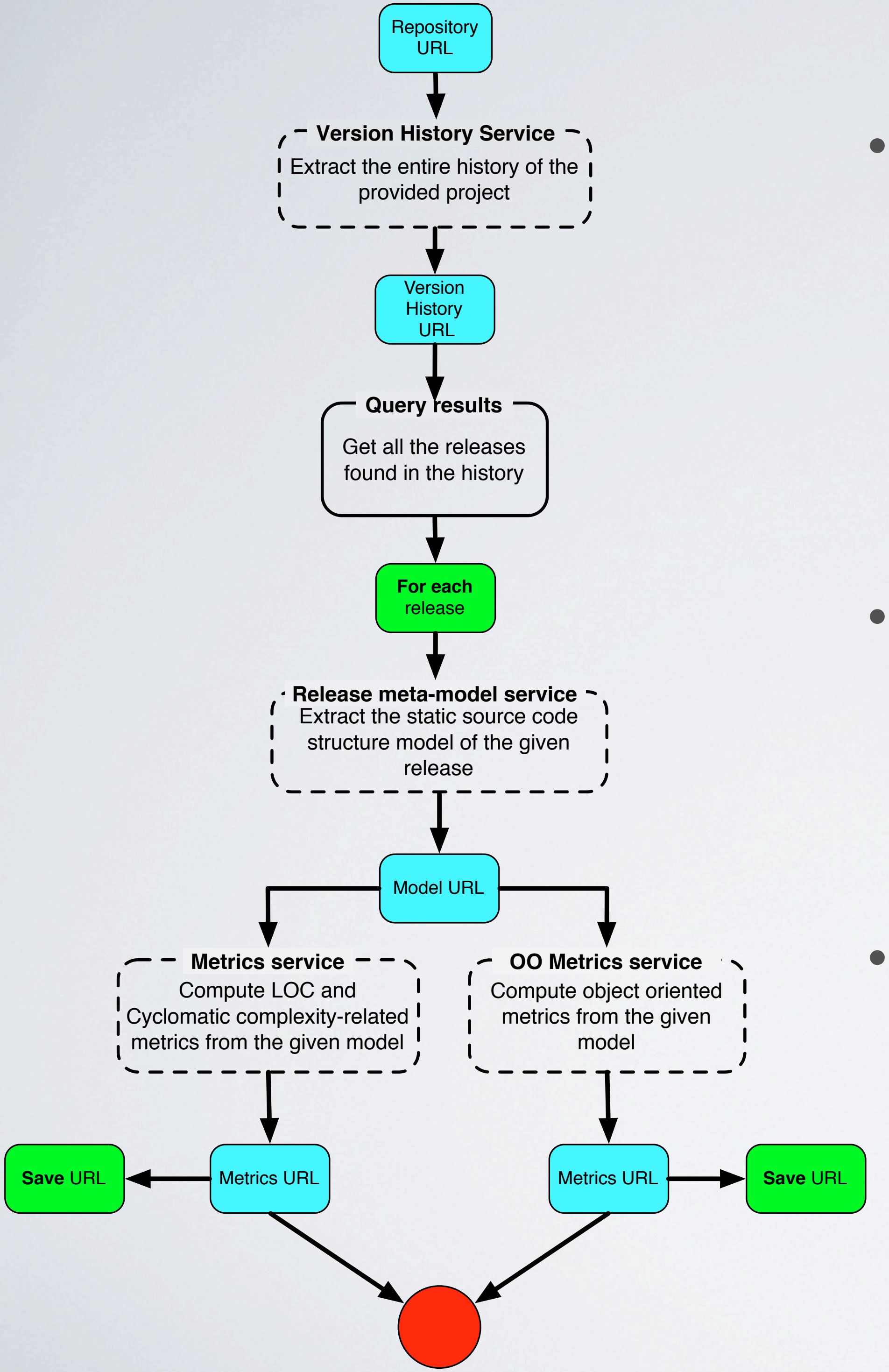
THE BENEFITS OF OWL ONTOLOGIES

- Clear semantics and precise syntax of input and output data
- Heterogenous domain ontologies can be semantically "linked" to each other by means of upper ontologies, describing general concepts across a wide range of domains
- Automatic reasoning and possibility to infer additional knowledge
- A standard, powerful query language already exists (SPARQL)
- No backwards compatibility problems upon data model extension

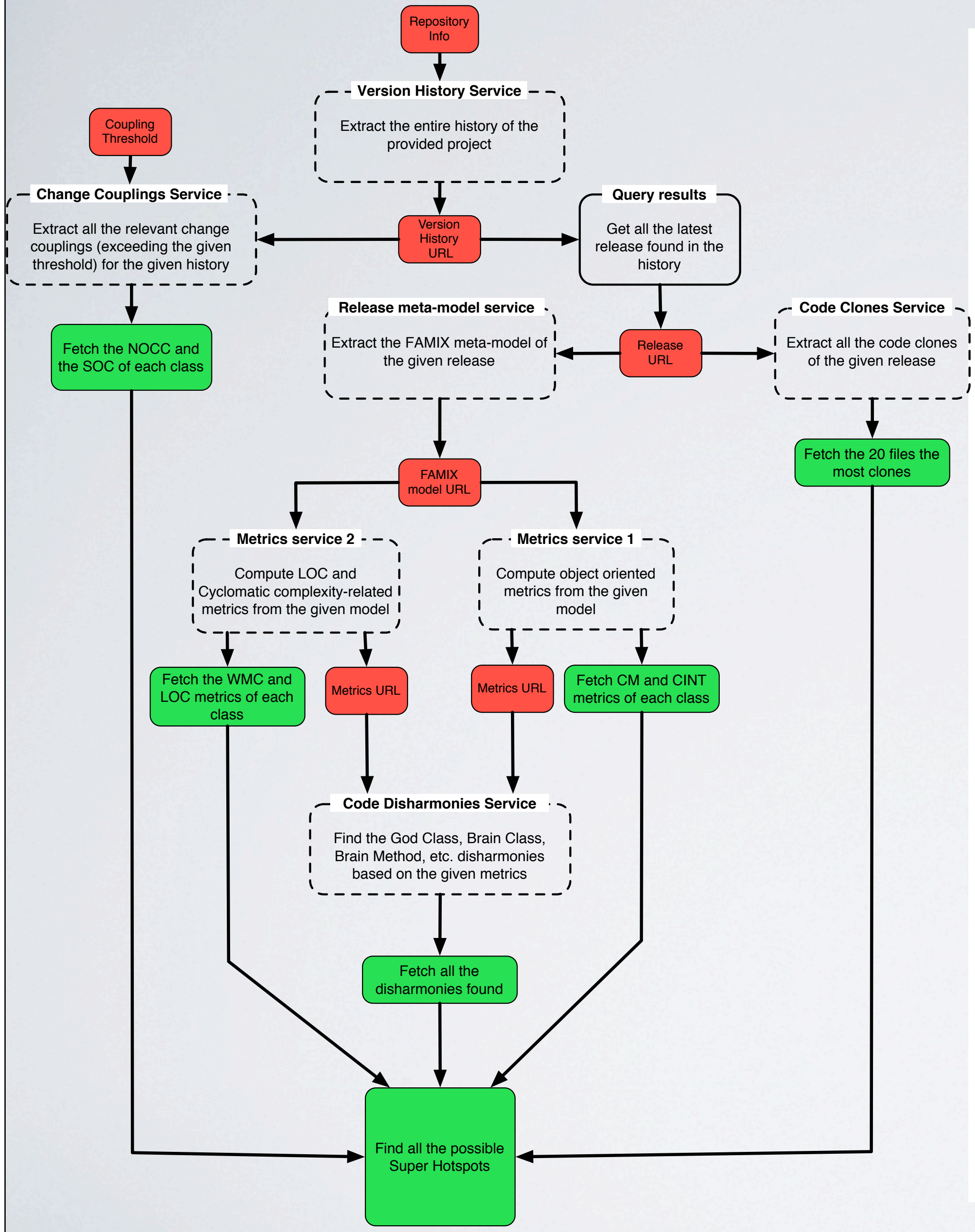
How can we (semi)-automatically compose these analyses to provide more complex higher-level analyses?

SCOLA

- **S**OFAS **Co**mposition **La**nguage
- A custom service composition language (based on WS-BPEL) to can be used by outsiders
- Existing languages are mainly for classic web services and add unneeded complexity
- The way SOFAS services behave and can be used allowed us to simplify the language


Repository URL
Version History Service
Extract the entire history of the provided project
Version History URL
Query results
Get all the releases found in the history
For each release
Release meta-model service
Extract the static source code structure model of the given release
Model URL
Metrics service
Compute LOC and Cyclomatic complexity-related metrics from the given model
OO Metrics service
Compute object oriented metrics from the given model
Save URL
Metrics URL
Metrics URL
Save URL

- **variables**
  - ➡ to define data passed to services by the user
  - ➡ to maintain information between service calls
  - ➡ to pass data between services
- **primitive activities**
  - ➡ to interact with services: *invoke* and *query*
  - ➡ to perform simple atomic actions: *exit*, *empty* and *save*
- **structured activities**
  - ➡ to define the execution flow: *sequence*, *flow*, *for each*, *if*, *link*.
  - ➡ can be nested and composed with each other



Repository Info

Version History Service

Extract the entire history of the provided project

Coupling Threshold

Change Couplings Service

Extract all the relevant change couplings (exceeding the given threshold) for the given history

Version History URL

Query results

Get all the latest release found in the history

Fetch the NOCC and the SOC of each class

Release meta-model service

Extract the FAMIX meta-model of the given release

Release URL

Code Clones Service

Extract all the code clones of the given release

Fetch the 20 files the most clones

FAMIX model URL

Metrics service 2

Compute LOC and Cyclomatic complexity-related metrics from the given model

Metrics service 1

Compute object oriented metrics from the given model

Fetch the WMC and LOC metrics of each class

Metrics URL

Metrics URL

Fetch CM and CINT metrics of each class

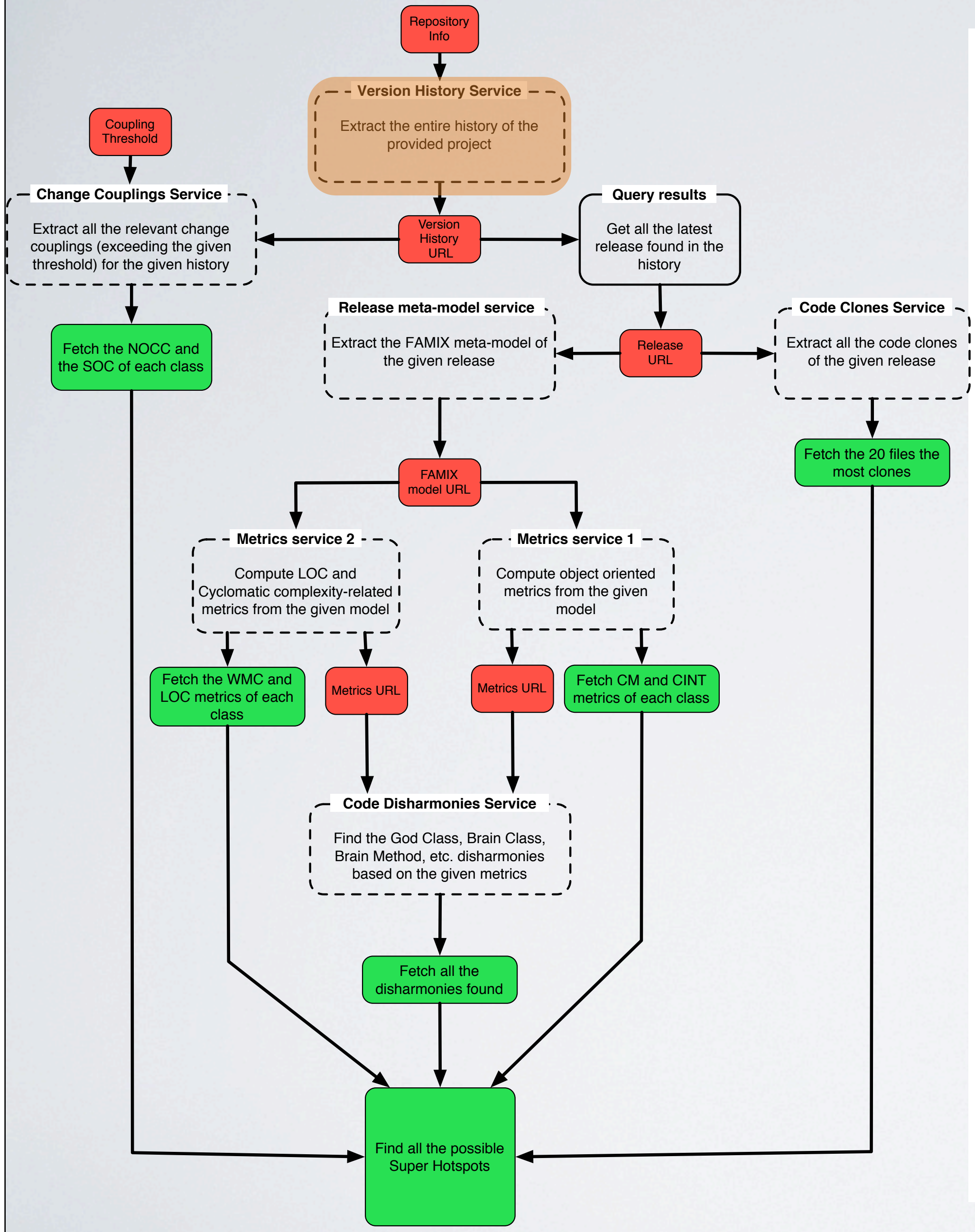
Code Disharmonies Service

Find the God Class, Brain Class, Brain Method, etc. disharmonies based on the given metrics

Fetch all the disharmonies found

Find all the possible Super Hotspots

```
<links>
     <link name="versionHistory_to_changeCoupling"/>
     <link name="versionHistory_to_fetchReleaseQuery"/>
</links>
<sequence>
     <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractVersionHistory"
          inputVariable="repositoryData"
          outputVariable="historyUrl">
        <sources>
            <source linkName="versionHistory_to_changeCoupling"/>
            <source linkName="versionHistory_to_fetchReleaseQuery"/>
        </sources>
    </invoke>
    <assign>
        <copy>
            <from>$historyUrl</from>
            <to>$changeCouplingInput/customerInfo</to>
        </copy>
        <copy>
            <from>$historyUrl</from>
            <to>$inputUrl</to>
        </copy>
    </assign>
    <flow>
        <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractChangeCoupling
            inputVariable="changeCouplingInput">
            <targets>
                <target linkName="versionHistory_to_changeCoupling"/>
            </targets>
        </invoke>
        <query sparqlQuery="SELECT ?x WHERE ?x rdf:type v:Release . . . . . ."
             inputVariable="inputUrl"
             outputVariable="lastReleaseUrl">
             <targets>
                 <target linkName="versionHistory_to_fetchReleaseQuery"/>
             </targets>
        </query>
    </flow>
</sequence>
```



Repository Info

Version History Service

Extract the entire history of the provided project

Coupling Threshold

Change Couplings Service

Extract all the relevant change couplings (exceeding the given threshold) for the given history

Version History URL

Query results

Get all the latest release found in the history

Fetch the NOCC and the SOC of each class

Release meta-model service

Extract the FAMIX meta-model of the given release

Release URL

Code Clones Service

Extract all the code clones of the given release

Fetch the 20 files the most clones

FAMIX model URL

Metrics service 2

Compute LOC and Cyclomatic complexity-related metrics from the given model

Metrics service 1

Compute object oriented metrics from the given model

Fetch the WMC and LOC metrics of each class

Metrics URL

Metrics URL

Fetch CM and CINT metrics of each class

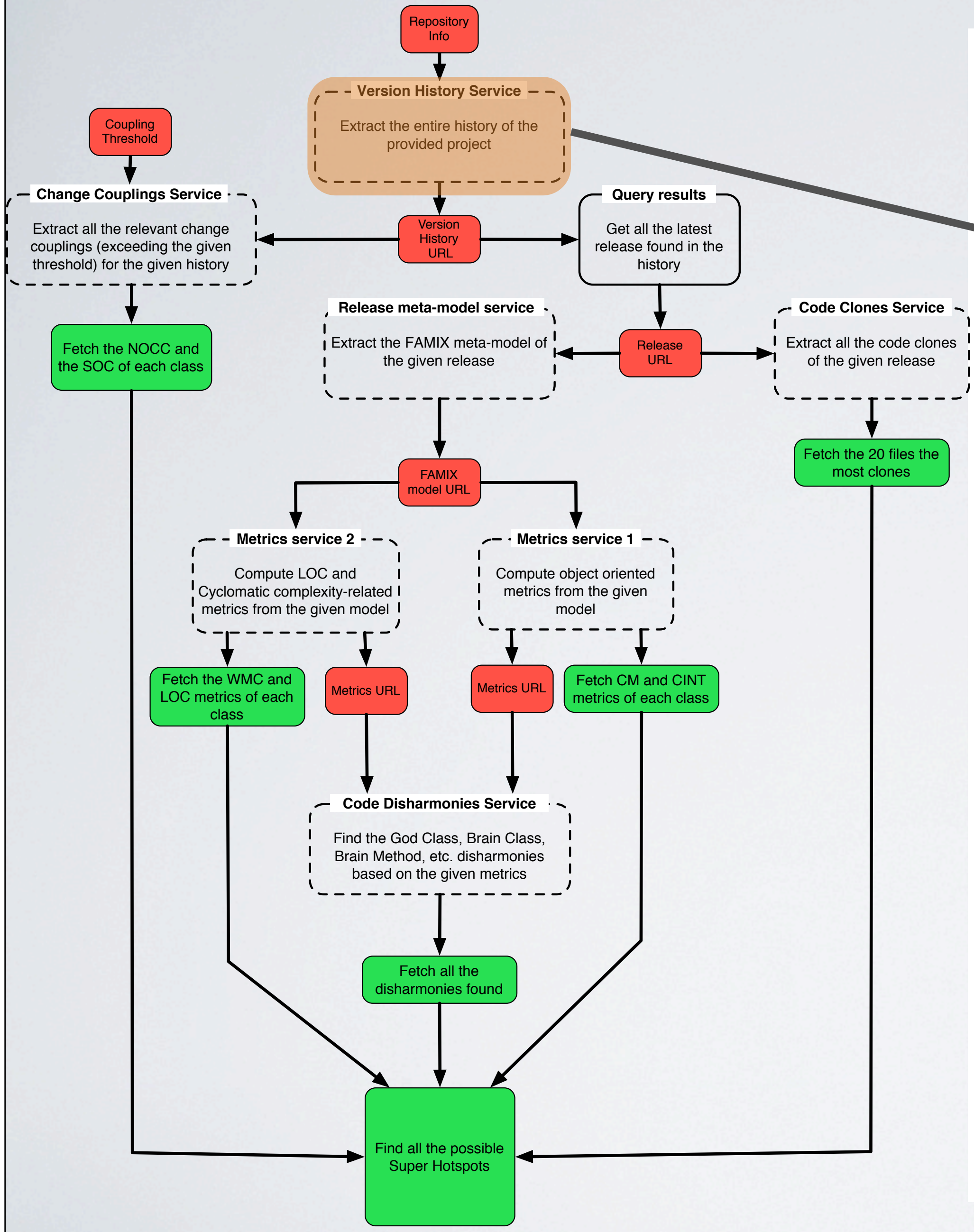
Code Disharmonies Service

Find the God Class, Brain Class, Brain Method, etc. disharmonies based on the given metrics

Fetch all the disharmonies found

Find all the possible Super Hotspots

```
<links>
        <link name="versionHistory_to_changeCoupling"/>
        <link name="versionHistory_to_fetchReleaseQuery"/>
</links>
<sequence>
        <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractVersionHistory"
               inputVariable="repositoryData"
               outputVariable="historyUrl">
            <sources>
                  <source linkName="versionHistory_to_changeCoupling"/>
                  <source linkName="versionHistory_to_fetchReleaseQuery"/>
            </sources>
      </invoke>
      <assign>
            <copy>
                  <from>$historyUrl</from>
                  <to>$changeCouplingInput/customerInfo</to>
            </copy>
            <copy>
                  <from>$historyUrl</from>
                  <to>$inputUrl</to>
            </copy>
      </assign>
      <flow>
            <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractChangeCoupling
                  inputVariable="changeCouplingInput">
                  <targets>
                        <target linkName="versionHistory_to_changeCoupling"/>
                  </targets>
            </invoke>
            <query sparqlQuery="SELECT ?x WHERE ?x rdf:type v:Release . . . . . ."
                   inputVariable="inputUrl"
                   outputVariable="lastReleaseUrl">
                   <targets>
                        <target linkName="versionHistory_to_fetchReleaseQuery"/>
                   </targets>
            </query>
      </flow>
</sequence>
```


Repository Info

Version History Service

Extract the entire history of the provided project

Coupling Threshold

Change Couplings Service

Extract all the relevant change couplings (exceeding the given threshold) for the given history

Version History URL

Query results

Get all the latest release found in the history

Fetch the NOCC and the SOC of each class

Release meta-model service

Extract the FAMIX meta-model of the given release

Release URL

Code Clones Service

Extract all the code clones of the given release

Fetch the 20 files the most clones

FAMIX model URL

Metrics service 2

Compute LOC and Cyclomatic complexity-related metrics from the given model

Metrics service 1

Compute object oriented metrics from the given model

Fetch the WMC and LOC metrics of each class

Metrics URL

Metrics URL

Fetch CM and CINT metrics of each class

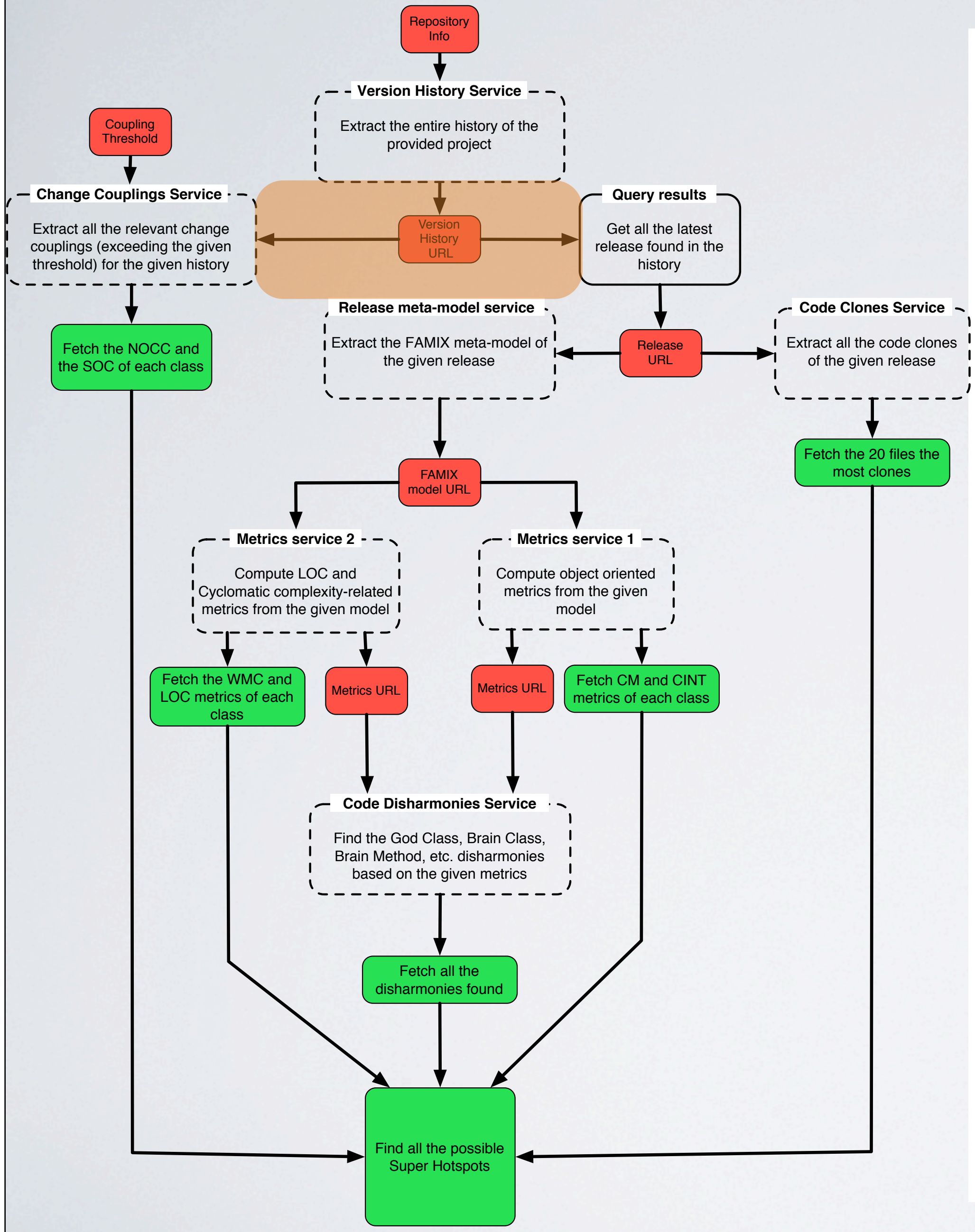
Code Disharmonies Service

Find the God Class, Brain Class, Brain Method, etc. disharmonies based on the given metrics

Fetch all the disharmonies found

Find all the possible Super Hotspots

```
<links>
    <link name="versionHistory_to_changeCoupling"/>
    <link name="versionHistory_to_fetchReleaseQuery"/>
</links>
<sequence>
    <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractVersionHistory"
        inputVariable="repositoryData"
        outputVariable="historyUrl">
      <sources>
         <source linkName="versionHistory_to_changeCoupling"/>
         <source linkName="versionHistory_to_fetchReleaseQuery"/>
      </sources>
   </invoke>
   <assign>
      <copy>
         <from>$historyUrl</from>
         <to>$changeCouplingInput/customerInfo</to>
      </copy>
      <copy>
         <from>$historyUrl</from>
         <to>$inputUrl</to>
      </copy>
   </assign>
   <flow>
      <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractChangeCoupling
          inputVariable="changeCouplingInput">
          <targets>
             <target linkName="versionHistory_to_changeCoupling"/>
          </targets>
      </invoke>
      <query sparqlQuery="SELECT ?x WHERE ?x rdf:type v:Release . . . . . ."
          inputVariable="inputUrl"
          outputVariable="lastReleaseUrl">
          <targets>
             <target linkName="versionHistory_to_fetchReleaseQuery"/>
          </targets>
      </query>
   </flow>
</sequence>
```



Repository Info

Version History Service

Extract the entire history of the provided project

Coupling Threshold

Change Couplings Service

Extract all the relevant change couplings (exceeding the given threshold) for the given history

Version History URL

Query results

Get all the latest release found in the history

Fetch the NOCC and the SOC of each class

Release meta-model service

Extract the FAMIX meta-model of the given release

Release URL

Code Clones Service

Extract all the code clones of the given release

Fetch the 20 files the most clones

FAMIX model URL

Metrics service 2

Compute LOC and Cyclomatic complexity-related metrics from the given model

Metrics service 1

Compute object oriented metrics from the given model

Fetch the WMC and LOC metrics of each class

Metrics URL

Metrics URL

Fetch CM and CINT metrics of each class

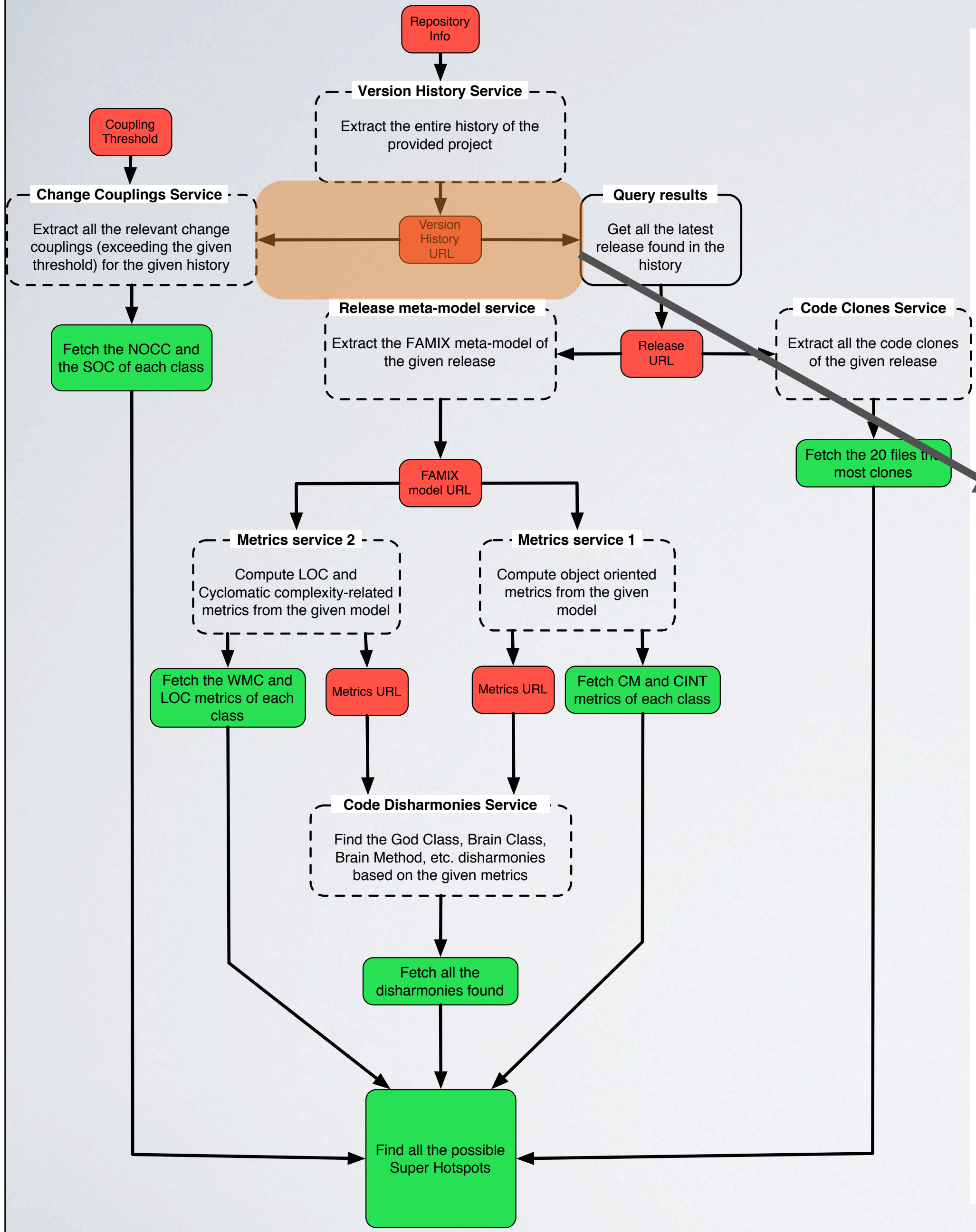
Code Disharmonies Service

Find the God Class, Brain Class, Brain Method, etc. disharmonies based on the given metrics

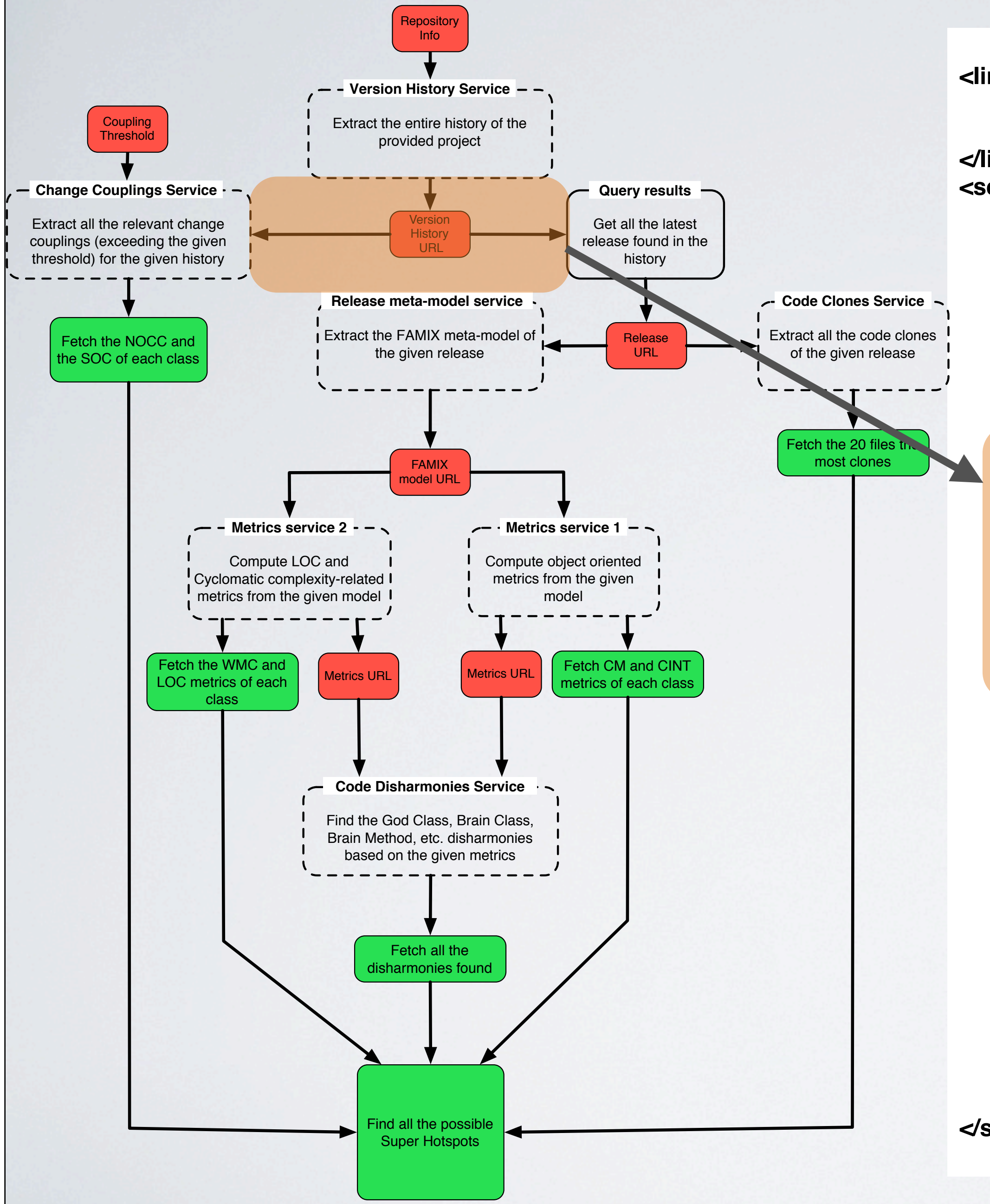
Fetch all the disharmonies found

Find all the possible Super Hotspots

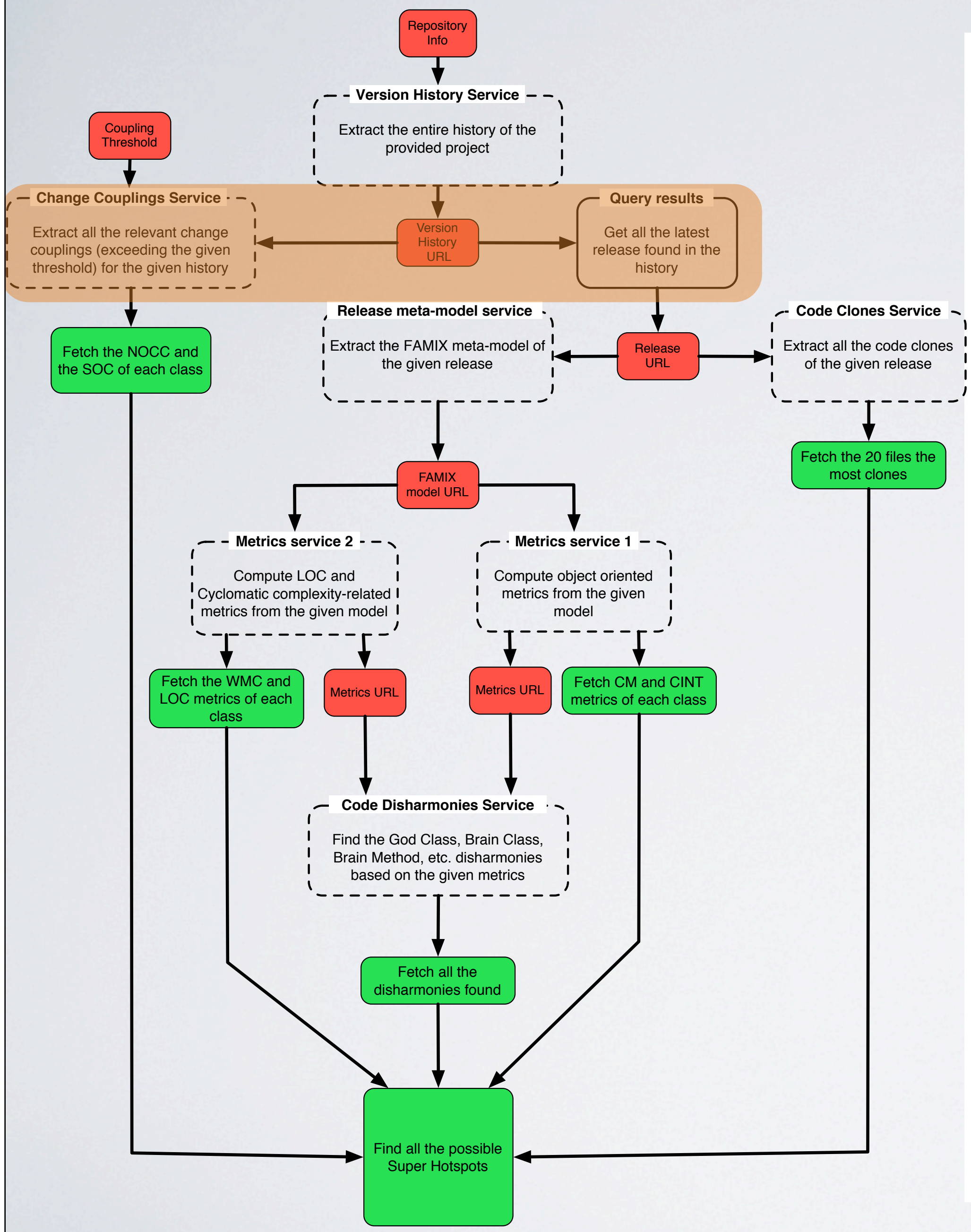
```
<links>
    <link name="versionHistory_to_changeCoupling"/>
    <link name="versionHistory_to_fetchReleaseQuery"/>
</links>
<sequence>
    <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractVersionHistory"
        inputVariable="repositoryData"
        outputVariable="historyUrl">
      <sources>
         <source linkName="versionHistory_to_changeCoupling"/>
         <source linkName="versionHistory_to_fetchReleaseQuery"/>
      </sources>
   </invoke>
   <assign>
      <copy>
         <from>$historyUrl</from>
         <to>$changeCouplingInput/customerInfo</to>
      </copy>
      <copy>
         <from>$historyUrl</from>
         <to>$inputUrl</to>
      </copy>
   </assign>
   <flow>
      <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractChangeCoupling
         inputVariable="changeCouplingInput">
         <targets>
            <target linkName="versionHistory_to_changeCoupling"/>
         </targets>
      </invoke>
      <query sparqlQuery="SELECT ?x WHERE ?x rdf:type v:Release . . . . . ."
         inputVariable="inputUrl"
         outputVariable="lastReleaseUrl">
         <targets>
            <target linkName="versionHistory_to_fetchReleaseQuery"/>
         </targets>
      </query>
   </flow>
</sequence>
```


Repository Info
Version History Service
Extract the entire history of the provided project
Coupling Threshold
Change Couplings Service
Extract all the relevant change couplings (exceeding the given threshold) for the given history
Version History URL
Query results
Get all the latest release found in the history
Release meta-model service
Extract the FAMIX meta-model of the given release
Release URL
Code Clones Service
Extract all the code clones of the given release
Fetch the NOCC and the SOC of each class
Fetch the 20 files th most clones
FAMIX model URL
Metrics service 2
Compute LOC and Cyclomatic complexity-related metrics from the given model
Metrics service 1
Compute object oriented metrics from the given model
Fetch the WMC and LOC metrics of each class
Metrics URL
Metrics URL
Fetch CM and CINT metrics of each class
Code Disharmonies Service
Find the God Class, Brain Class, Brain Method, etc. disharmonies based on the given metrics
Fetch all the disharmonies found
Find all the possible Super Hotspots

```
<links>
      <link name="versionHistory_to_changeCoupling"/>
      <link name="versionHistory_to_fetchReleaseQuery"/>
</links>
<sequence>
      <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractVersionHistory"
           inputVariable="repositoryData"
           outputVariable="historyUrl">
         <sources>
             <source linkName="versionHistory_to_changeCoupling"/>
             <source linkName="versionHistory_to_fetchReleaseQuery"/>
         </sources>
    </invoke>
    <assign>
       <copy>
           <from>$historyUrl</from>
           <to>$changeCouplingInput/customerInfo</to>
        </copy>
        <copy>
           <from>$historyUrl</from>
           <to>$inputUrl</to>
        </copy>
    </assign>
    <flow>
        <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractChangeCoupling
             inputVariable="changeCouplingInput">
             <targets>
                 <target linkName="versionHistory_to_changeCoupling"/>
             </targets>
        </invoke>
        <query sparqlQuery="SELECT ?x WHERE ?x rdf:type v:Release . . . . . ."
              inputVariable="inputUrl"
              outputVariable="lastReleaseUrl">
              <targets>
                  <target linkName="versionHistory_to_fetchReleaseQuery"/>
              </targets>
        </query>
    </flow>
</sequence>
```

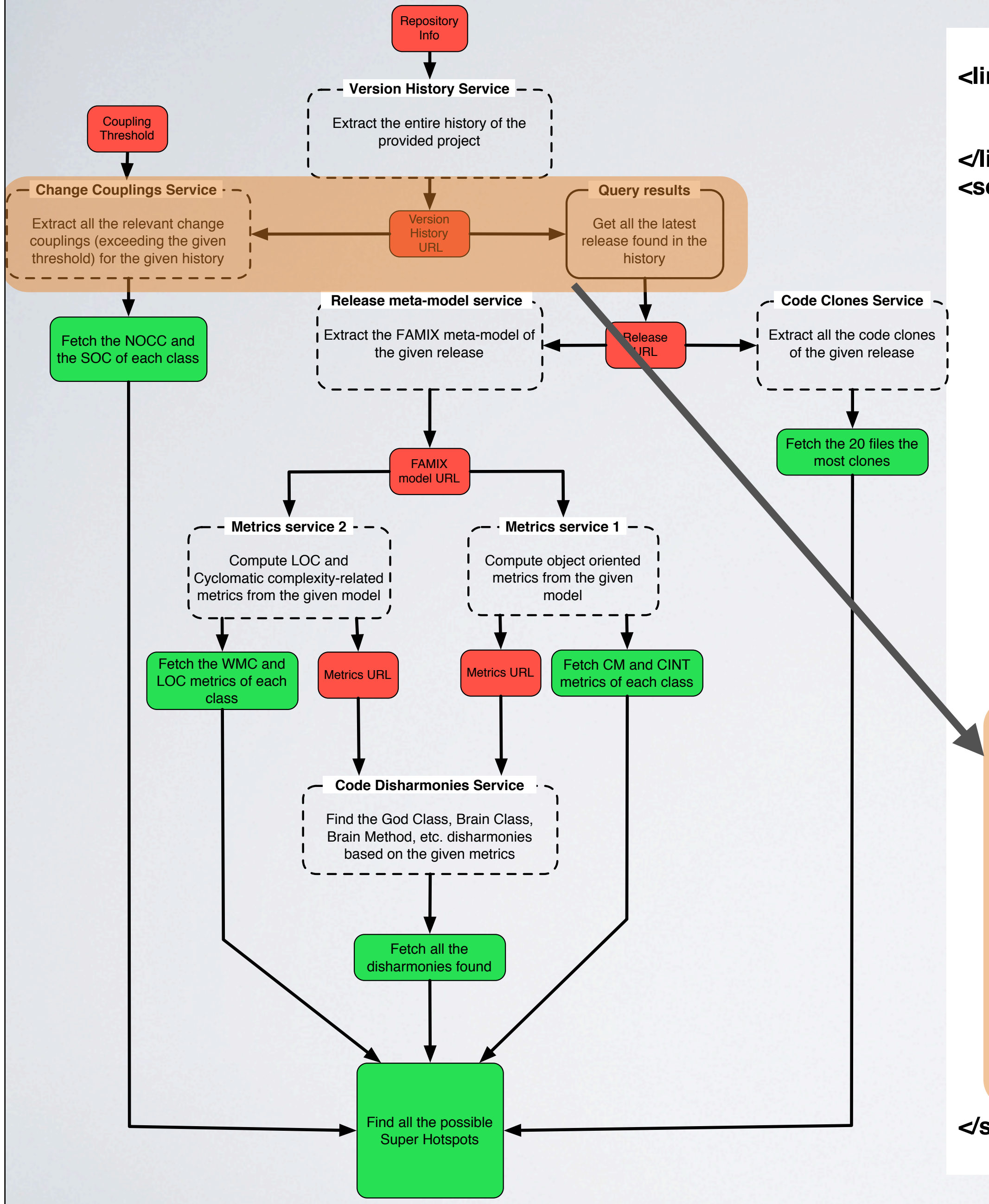

Repository Info
Version History Service
Extract the entire history of the provided project
Coupling Threshold
Change Couplings Service
Extract all the relevant change couplings (exceeding the given threshold) for the given history
Version History URL
Query results
Get all the latest release found in the history
Release meta-model service
Extract the FAMIX meta-model of the given release
Release URL
Code Clones Service
Extract all the code clones of the given release
Fetch the NOCC and the SOC of each class
Fetch the 20 files the most clones
FAMIX model URL
Metrics service 2
Compute LOC and Cyclomatic complexity-related metrics from the given model
Metrics service 1
Compute object oriented metrics from the given model
Fetch the WMC and LOC metrics of each class
Metrics URL
Metrics URL
Fetch CM and CINT metrics of each class
Code Disharmonies Service
Find the God Class, Brain Class, Brain Method, etc. disharmonies based on the given metrics
Fetch all the disharmonies found
Find all the possible Super Hotspots

```
<links>
        <link name="versionHistory_to_changeCoupling"/>
        <link name="versionHistory_to_fetchReleaseQuery"/>
</links>
<sequence>
        <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractVersionHistory"
            inputVariable="repositoryData"
            outputVariable="historyUrl">
            <sources>
                <source linkName="versionHistory_to_changeCoupling"/>
                <source linkName="versionHistory_to_fetchReleaseQuery"/>
            </sources>
        </invoke>
        <assign>
            <copy>
                <from>$historyUrl</from>
                <to>$changeCouplingInput/customerInfo</to>
            </copy>
            <copy>
                <from>$historyUrl</from>
                <to>$inputUrl</to>
            </copy>
        </assign>
        <flow>
            <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractChangeCoupling
                inputVariable="changeCouplingInput">
                <targets>
                    <target linkName="versionHistory_to_changeCoupling"/>
                </targets>
            </invoke>
            <query sparqlQuery="SELECT ?x WHERE ?x rdf:type v:Release . . . . . ."
                inputVariable="inputUrl"
                outputVariable="lastReleaseUrl">
                <targets>
                    <target linkName="versionHistory_to_fetchReleaseQuery"/>
                </targets>
            </query>
        </flow>
</sequence>
```

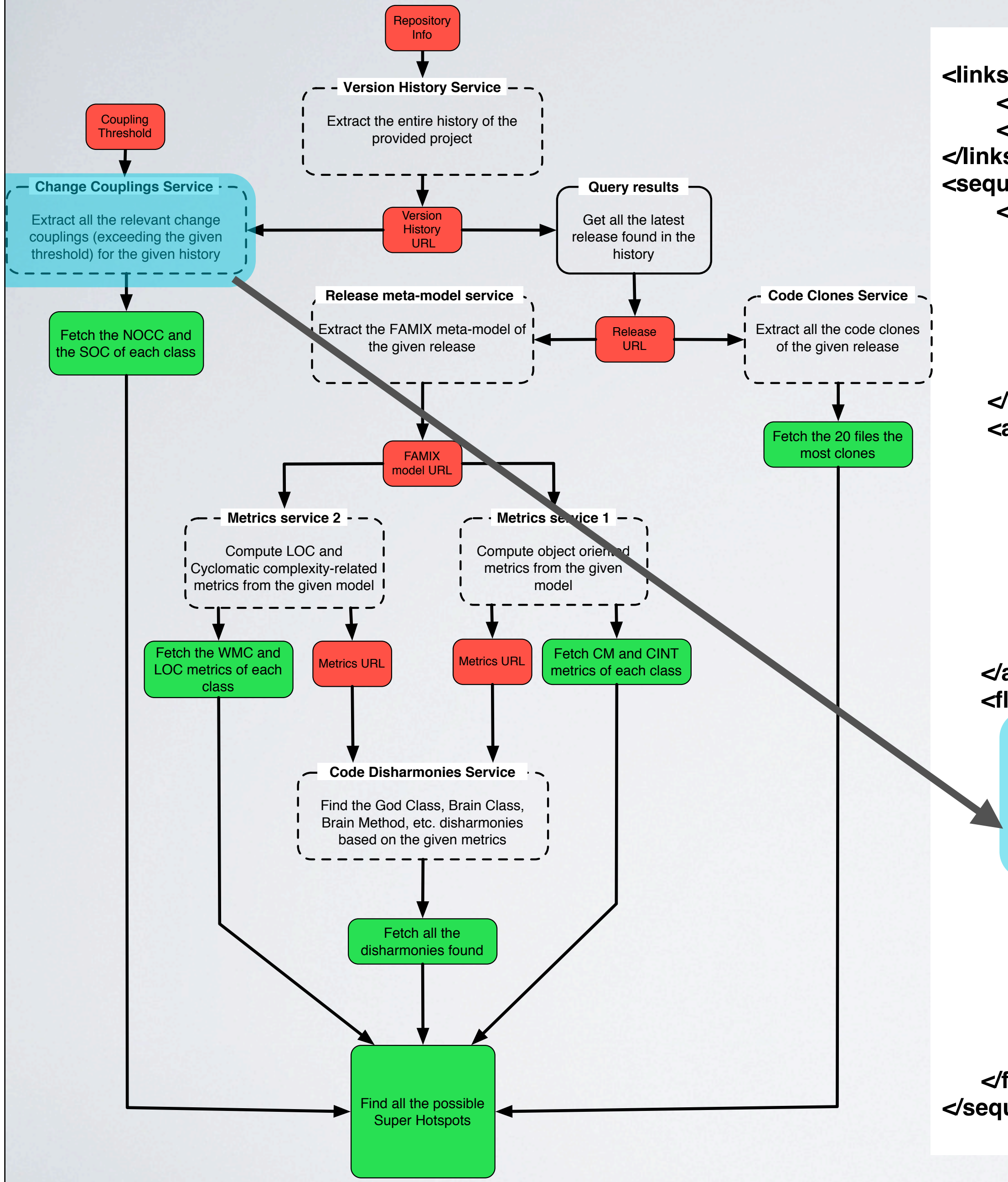


Repository Info
Version History Service
Extract the entire history of the provided project
Coupling Threshold
Change Couplings Service
Extract all the relevant change couplings (exceeding the given threshold) for the given history
Version History URL
Query results
Get all the latest release found in the history
Fetch the NOCC and the SOC of each class
Release meta-model service
Extract the FAMIX meta-model of the given release
Release URL
Code Clones Service
Extract all the code clones of the given release
Fetch the 20 files the most clones
FAMIX model URL
Metrics service 2
Compute LOC and Cyclomatic complexity-related metrics from the given model
Metrics service 1
Compute object oriented metrics from the given model
Fetch the WMC and LOC metrics of each class
Metrics URL
Metrics URL
Fetch CM and CINT metrics of each class
Code Disharmonies Service
Find the God Class, Brain Class, Brain Method, etc. disharmonies based on the given metrics
Fetch all the disharmonies found
Find all the possible Super Hotspots

```
<links>
        <link name="versionHistory_to_changeCoupling"/>
        <link name="versionHistory_to_fetchReleaseQuery"/>
</links>
<sequence>
        <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractVersionHistory"
               inputVariable="repositoryData"
               outputVariable="historyUrl">
            <sources>
                  <source linkName="versionHistory_to_changeCoupling"/>
                  <source linkName="versionHistory_to_fetchReleaseQuery"/>
            </sources>
      </invoke>
      <assign>
           <copy>
                 <from>$historyUrl</from>
                 <to>$changeCouplingInput/customerInfo</to>
           </copy>
           <copy>
                 <from>$historyUrl</from>
                 <to>$inputUrl</to>
           </copy>
     </assign>
     <flow>
           <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractChangeCoupling
                 inputVariable="changeCouplingInput">
                 <targets>
                      <target linkName="versionHistory_to_changeCoupling"/>
                 </targets>
           </invoke>
           <query sparqlQuery="SELECT ?x WHERE ?x rdf:type v:Release . . . . . ."
                   inputVariable="inputUrl"
                   outputVariable="lastReleaseUrl">
                  <targets>
                       <target linkName="versionHistory_to_fetchReleaseQuery"/>
                  </targets>
           </query>
     </flow>
</sequence>
```


Repository Info
Version History Service
Extract the entire history of the provided project
Coupling Threshold
Change Couplings Service
Extract all the relevant change couplings (exceeding the given threshold) for the given history
Version History URL
Query results
Get all the latest release found in the history
Fetch the NOCC and the SOC of each class
Release meta-model service
Extract the FAMIX meta-model of the given release
Release URL
Code Clones Service
Extract all the code clones of the given release
Fetch the 20 files the most clones
FAMIX model URL
Metrics service 2
Compute LOC and Cyclomatic complexity-related metrics from the given model
Metrics service 1
Compute object oriented metrics from the given model
Fetch the WMC and LOC metrics of each class
Metrics URL
Metrics URL
Fetch CM and CINT metrics of each class
Code Disharmonies Service
Find the God Class, Brain Class, Brain Method, etc. disharmonies based on the given metrics
Fetch all the disharmonies found
Find all the possible Super Hotspots

```
<links>
        <link name="versionHistory_to_changeCoupling"/>
        <link name="versionHistory_to_fetchReleaseQuery"/>
</links>
<sequence>
        <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractVersionHistory"
               inputVariable="repositoryData"
               outputVariable="historyUrl">
            <sources>
                  <source linkName="versionHistory_to_changeCoupling"/>
                  <source linkName="versionHistory_to_fetchReleaseQuery"/>
            </sources>
      </invoke>
      <assign>
           <copy>
                <from>$historyUrl</from>
                <to>$changeCouplingInput/customerInfo</to>
            </copy>
            <copy>
                <from>$historyUrl</from>
                <to>$inputUrl</to>
            </copy>
      </assign>
      <flow>
            <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractChangeCoupling
                  inputVariable="changeCouplingInput">
                  <targets>
                        <target linkName="versionHistory_to_changeCoupling"/>
                  </targets>
            </invoke>
            <query sparqlQuery="SELECT ?x WHERE ?x rdf:type v:Release . . . . . ."
                   inputVariable="inputUrl"
                   outputVariable="lastReleaseUrl">
                   <targets>
                         <target linkName="versionHistory_to_fetchReleaseQuery"/>
                   </targets>
            </query>
      </flow>
</sequence>
```



Repository Info

Version History Service

Extract the entire history of the provided project

Coupling Threshold

Change Couplings Service

Extract all the relevant change couplings (exceeding the given threshold) for the given history

Version History URL

Query results

Get all the latest release found in the history

Release meta-model service

Extract the FAMIX meta-model of the given release

Release URL

Code Clones Service

Extract all the code clones of the given release

Fetch the NOCC and the SOC of each class

Fetch the 20 files the most clones

FAMIX model URL

Metrics service 2

Compute LOC and Cyclomatic complexity-related metrics from the given model

Metrics service 1

Compute object oriented metrics from the given model

Fetch the WMC and LOC metrics of each class

Metrics URL

Metrics URL

Fetch CM and CINT metrics of each class

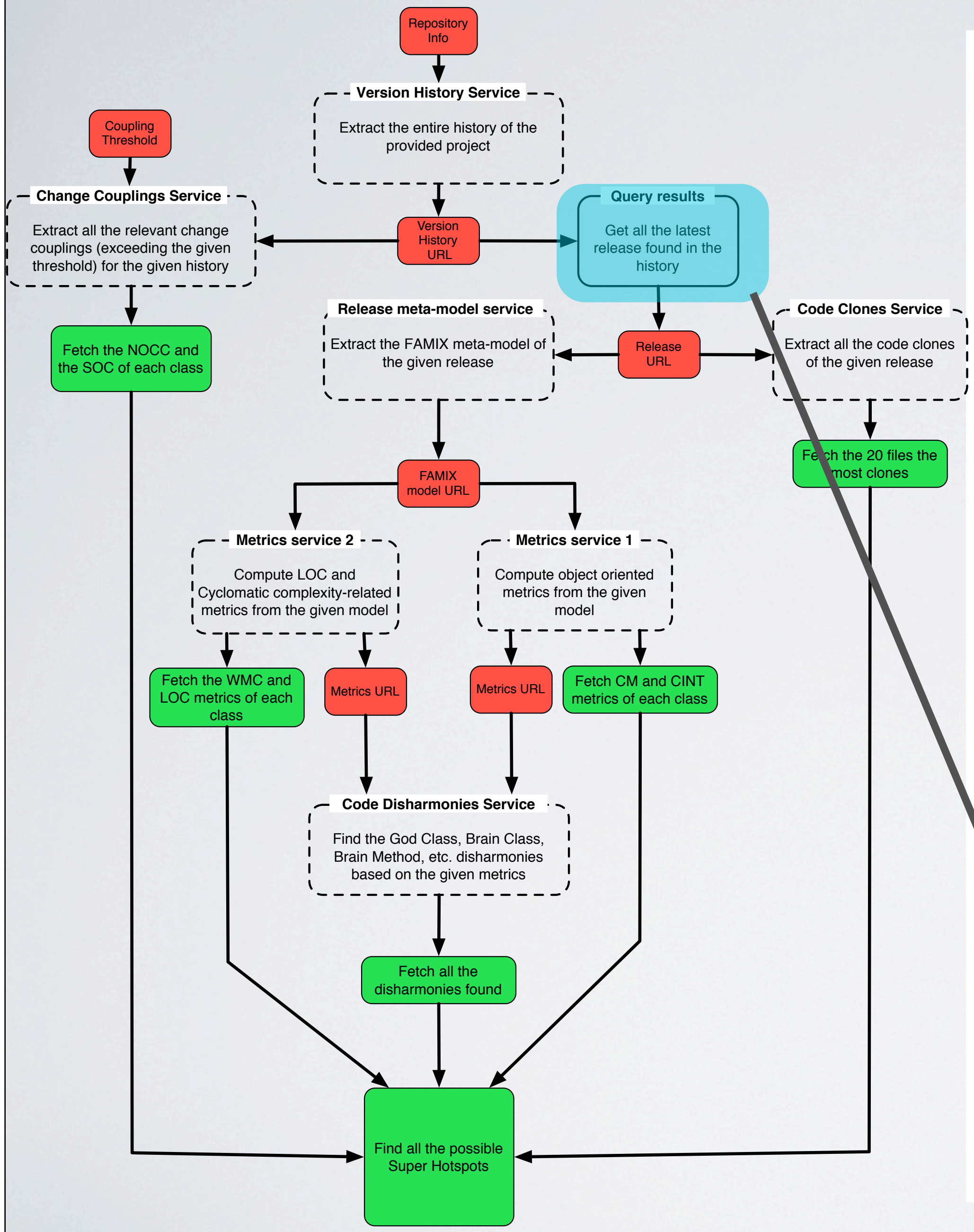
Code Disharmonies Service

Find the God Class, Brain Class, Brain Method, etc. disharmonies based on the given metrics

Fetch all the disharmonies found

Find all the possible Super Hotspots

```
<links>
    <link name="versionHistory_to_changeCoupling"/>
    <link name="versionHistory_to_fetchReleaseQuery"/>
</links>
<sequence>
    <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractVersionHistory"
        inputVariable="repositoryData"
        outputVariable="historyUrl">
      <sources>
        <source linkName="versionHistory_to_changeCoupling"/>
        <source linkName="versionHistory_to_fetchReleaseQuery"/>
      </sources>
  </invoke>
  <assign>
      <copy>
        <from>$historyUrl</from>
        <to>$changeCouplingInput/customerInfo</to>
      </copy>
      <copy>
        <from>$historyUrl</from>
        <to>$inputUrl</to>
      </copy>
  </assign>
  <flow>
      <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractChangeCoupling
          inputVariable="changeCouplingInput">
          <targets>
            <target linkName="versionHistory_to_changeCoupling"/>
          </targets>
      </invoke>
      <query sparqlQuery="SELECT ?x WHERE ?x rdf:type v:Release . . . . . ."
          inputVariable="inputUrl"
          outputVariable="lastReleaseUrl">
          <targets>
            <target linkName="versionHistory_to_fetchReleaseQuery"/>
          </targets>
      </query>
  </flow>
</sequence>
```



Repository Info

Version History Service

Extract the entire history of the provided project

Coupling Threshold

Change Couplings Service

Extract all the relevant change couplings (exceeding the given threshold) for the given history

Version History URL

Query results

Get all the latest release found in the history

Fetch the NOCC and the SOC of each class

Release meta-model service

Extract the FAMIX meta-model of the given release

Release URL

Code Clones Service

Extract all the code clones of the given release

Fetch the 20 files the most clones

FAMIX model URL

Metrics service 2

Compute LOC and Cyclomatic complexity-related metrics from the given model

Metrics service 1

Compute object oriented metrics from the given model

Fetch the WMC and LOC metrics of each class

Metrics URL

Metrics URL

Fetch CM and CINT metrics of each class

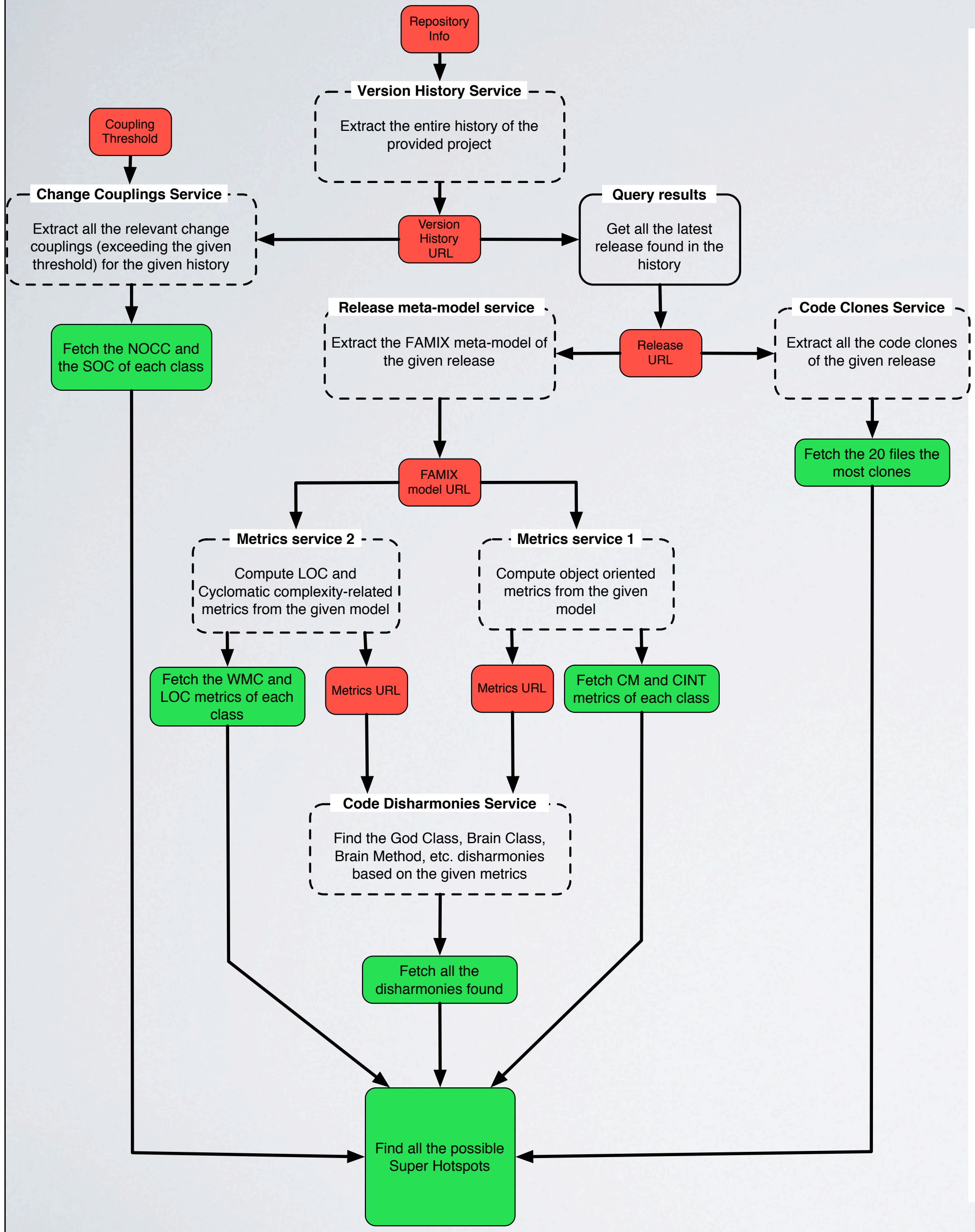
Code Disharmonies Service

Find the God Class, Brain Class, Brain Method, etc. disharmonies based on the given metrics

Fetch all the disharmonies found

Find all the possible Super Hotspots

```
<links>
    <link name="versionHistory_to_changeCoupling"/>
    <link name="versionHistory_to_fetchReleaseQuery"/>
</links>
<sequence>
    <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractVersionHistory"
        inputVariable="repositoryData"
        outputVariable="historyUrl">
      <sources>
         <source linkName="versionHistory_to_changeCoupling"/>
         <source linkName="versionHistory_to_fetchReleaseQuery"/>
      </sources>
   </invoke>
   <assign>
      <copy>
         <from>$historyUrl</from>
         <to>$changeCouplingInput/customerInfo</to>
      </copy>
      <copy>
         <from>$historyUrl</from>
         <to>$inputUrl</to>
      </copy>
   </assign>
   <flow>
      <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractChangeCoupling
         inputVariable="changeCouplingInput">
         <targets>
            <target linkName="versionHistory_to_changeCoupling"/>
         </targets>
      </invoke>
      <query sparqlQuery="SELECT ?x WHERE ?x rdf:type v:Release . . . . . ."
         inputVariable="inputUrl"
         outputVariable="lastReleaseUrl">
         <targets>
            <target linkName="versionHistory_to_fetchReleaseQuery"/>
         </targets>
      </query>
   </flow>
</sequence>
```



Repository Info

Version History Service

Extract the entire history of the provided project

Coupling Threshold

Change Couplings Service

Extract all the relevant change couplings (exceeding the given threshold) for the given history

Version History URL

Query results

Get all the latest release found in the history

Fetch the NOCC and the SOC of each class

Release meta-model service

Extract the FAMIX meta-model of the given release

Release URL

Code Clones Service

Extract all the code clones of the given release

Fetch the 20 files the most clones

FAMIX model URL

Metrics service 2

Compute LOC and Cyclomatic complexity-related metrics from the given model

Metrics service 1

Compute object oriented metrics from the given model

Fetch the WMC and LOC metrics of each class

Metrics URL

Metrics URL

Fetch CM and CINT metrics of each class

Code Disharmonies Service

Find the God Class, Brain Class, Brain Method, etc. disharmonies based on the given metrics

Fetch all the disharmonies found

Find all the possible Super Hotspots

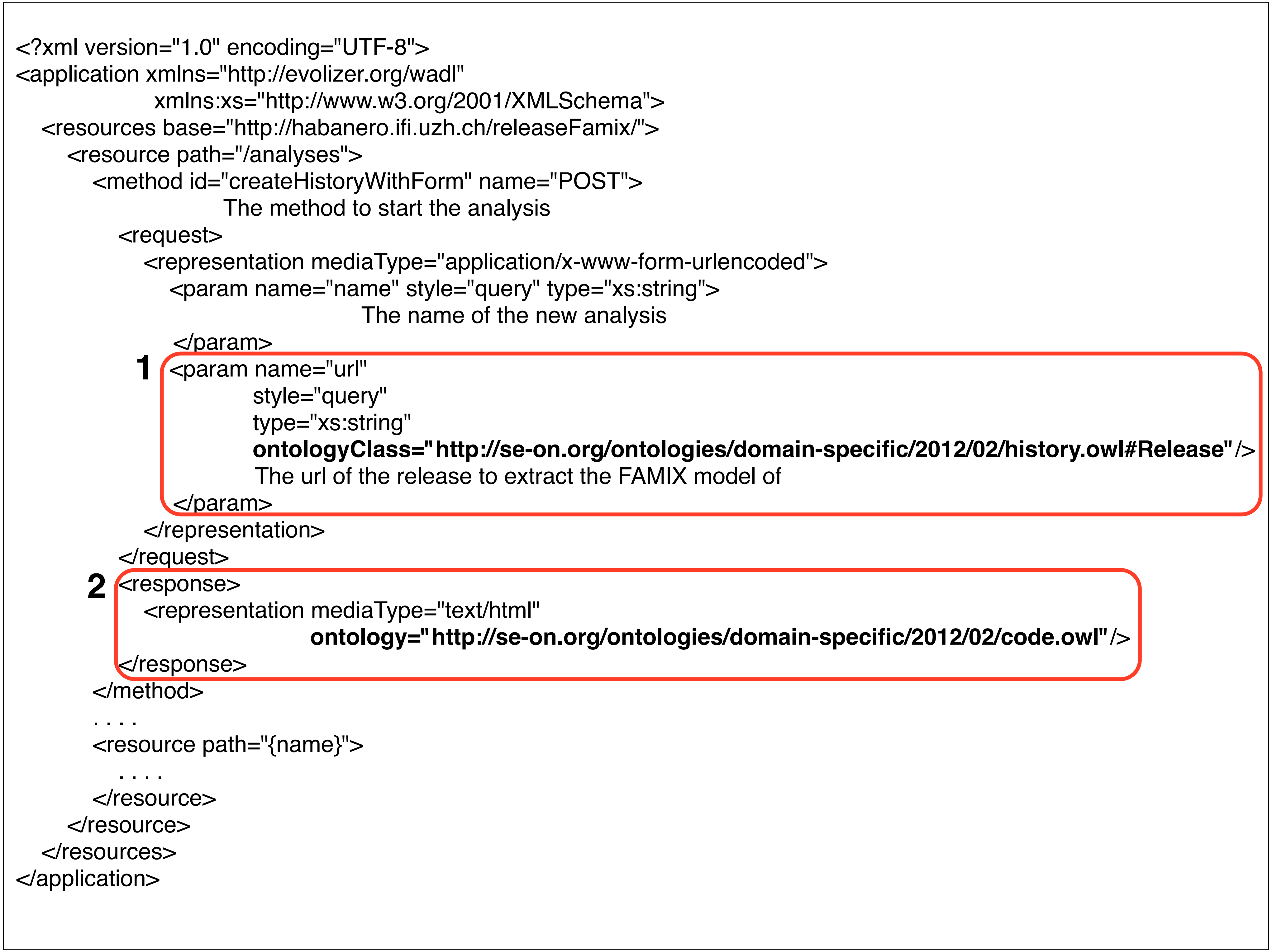
```
<links>
    <link name="versionHistory_to_changeCoupling"/>
    <link name="versionHistory_to_fetchReleaseQuery"/>
</links>
<sequence>
    <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractVersionHistory"
        inputVariable="repositoryData"
        outputVariable="historyUrl">
      <sources>
          <source linkName="versionHistory_to_changeCoupling"/>
          <source linkName="versionHistory_to_fetchReleaseQuery"/>
      </sources>
   </invoke>
   <assign>
      <copy>
         <from>$historyUrl</from>
         <to>$changeCouplingInput/customerInfo</to>
      </copy>
      <copy>
         <from>$historyUrl</from>
         <to>$inputUrl</to>
      </copy>
   </assign>
   <flow>
      <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractChangeCoupling
          inputVariable="changeCouplingInput">
        <targets>
            <target linkName="versionHistory_to_changeCoupling"/>
        </targets>
      </invoke>
      <query sparqlQuery="SELECT ?x WHERE ?x rdf:type v:Release . . . . . ."
           inputVariable="inputUrl"
           outputVariable="lastReleaseUrl">
        <targets>
           <target linkName="versionHistory_to_fetchReleaseQuery"/>
        </targets>
      </query>
   </flow>
</sequence>
```

WORKFLOW VALIDATION

- Syntactical correctness is checked and enforced using the constraints defined in the description (XML schema restrictions)

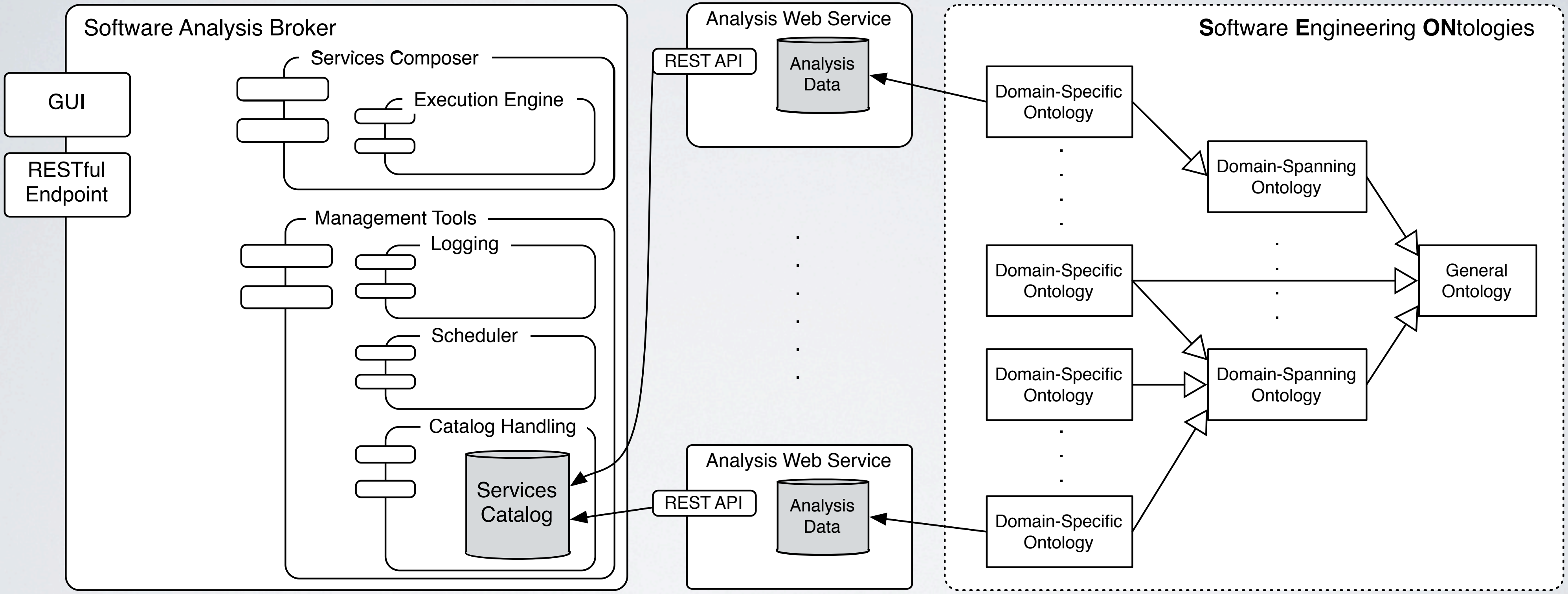
- Semantical correctness based on annotated WADL: any input and output of a service method, may be annotated with a URI to an ontology

➡ Very useful in guiding the composition

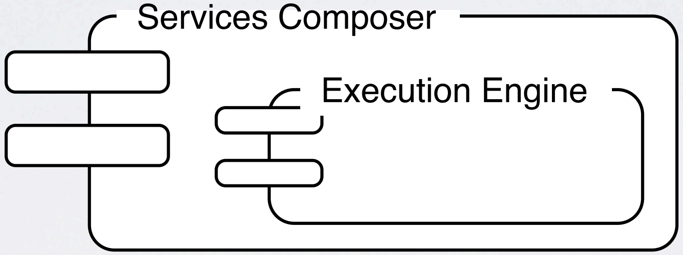


```
<?xml version="1.0" encoding="UTF-8">
<application xmlns="http://evolizer.org/wadl"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <resources base="http://habanero.ifi.uzh.ch/releaseFamix/">
      <resource path="/analyses">
         <method id="createHistoryWithForm" name="POST">
                        The method to start the analysis
            <request>
               <representation mediaType="application/x-www-form-urlencoded">
                  <param name="name" style="query" type="xs:string">
                                    The name of the new analysis
                  </param>
               1 <param name="url"
                            style="query"
                            type="xs:string"
                            ontologyClass="http://se-on.org/ontologies/domain-specific/2012/02/history.owl#Release" />
                            The url of the release to extract the FAMIX model of
                  </param>
               </representation>
            </request>
         2 <response>
               <representation mediaType="text/html"
                              ontology="http://se-on.org/ontologies/domain-specific/2012/02/code.owl" />
            </response>
         </method>
         . . . .
         <resource path="{name}">
            . . . .
         </resource>
      </resource>
   </resources>
</application>
```

SCOLA IN PRACTICE

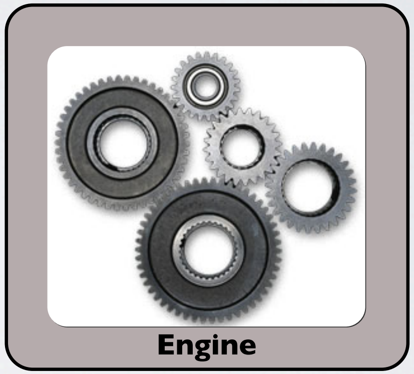


SCOLA IN PRACTICE



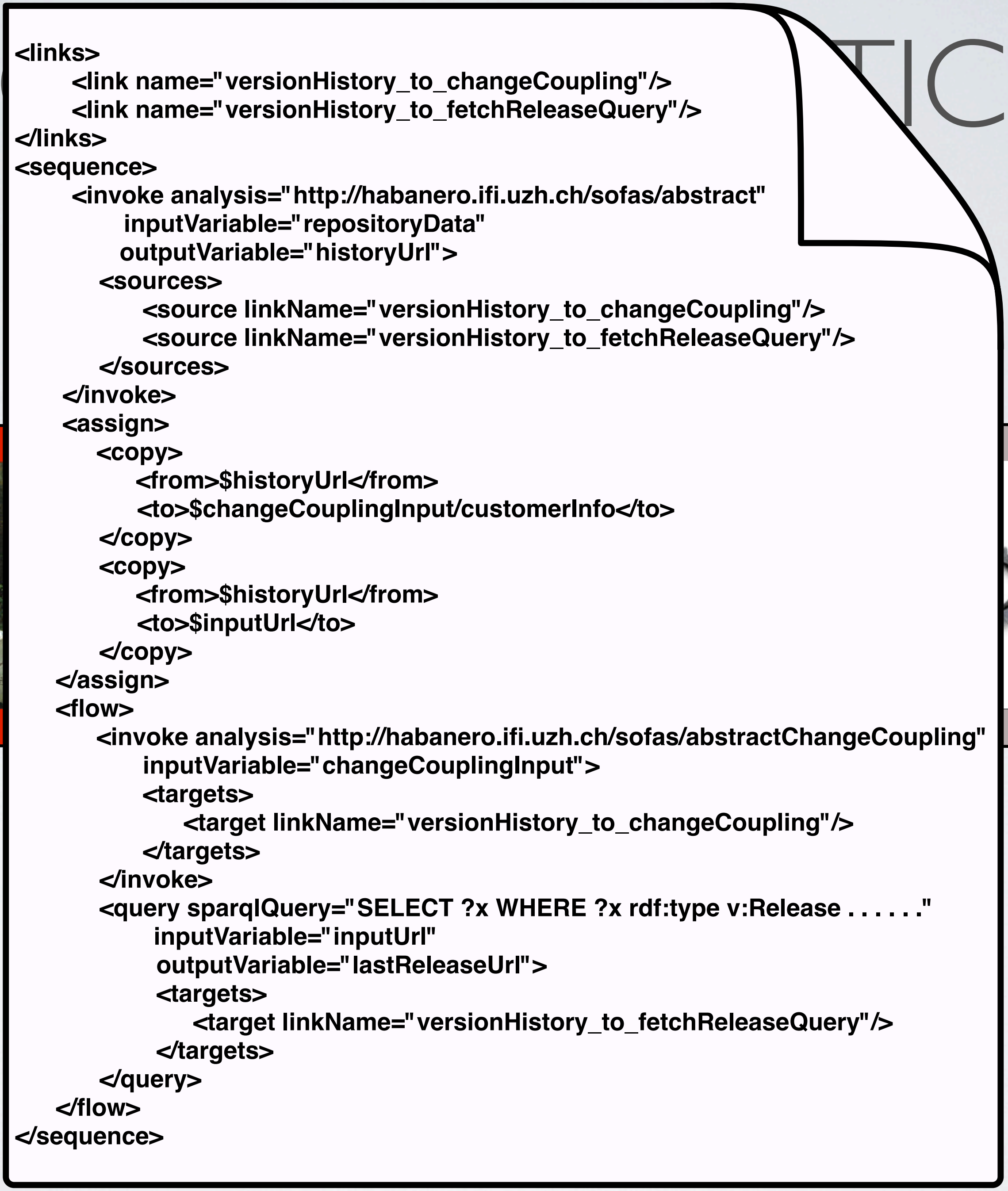
SCOLA IN PRACTICE

Composer



Engine

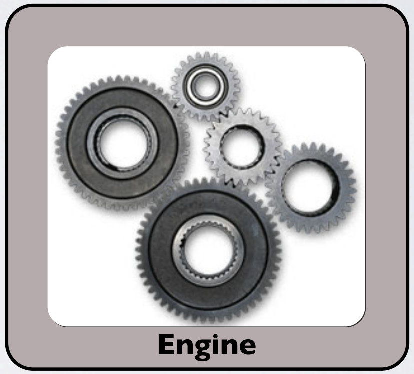
S TICE



```
<links>
      <link name="versionHistory_to_changeCoupling"/>
      <link name="versionHistory_to_fetchReleaseQuery"/>
</links>
<sequence>
      <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstract"
            inputVariable="repositoryData"
            outputVariable="historyUrl">
         <sources>
              <source linkName="versionHistory_to_changeCoupling"/>
              <source linkName="versionHistory_to_fetchReleaseQuery"/>
         </sources>
     </invoke>
     <assign>
         <copy>
              <from>$historyUrl</from>
              <to>$changeCouplingInput/customerInfo</to>
         </copy>
         <copy>
              <from>$historyUrl</from>
              <to>$inputUrl</to>
         </copy>
    </assign>
    <flow>
         <invoke analysis="http://habanero.ifi.uzh.ch/sofas/abstractChangeCoupling"
              inputVariable="changeCouplingInput">
              <targets>
                   <target linkName="versionHistory_to_changeCoupling"/>
              </targets>
         </invoke>
         <query sparqlQuery="SELECT ?x WHERE ?x rdf:type v:Release . . . . . ."
               inputVariable="inputUrl"
               outputVariable="lastReleaseUrl">
               <targets>
                    <target linkName="versionHistory_to_fetchReleaseQuery"/>
               </targets>
         </query>
    </flow>
</sequence>
```

SCOLA IN PRACTICE

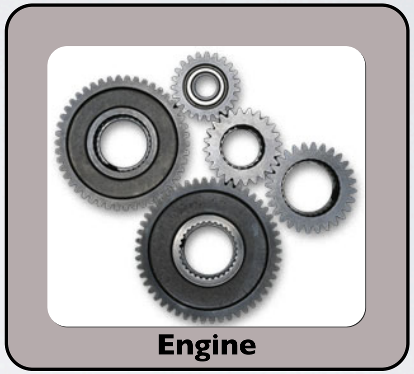
Composer



Engine

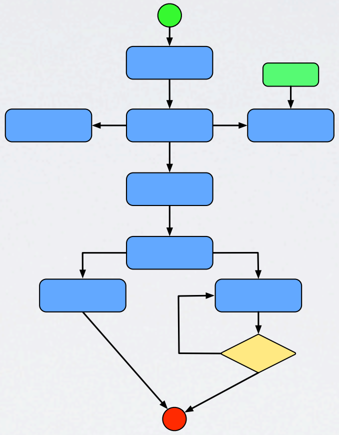
SCOLA IN PRACTICE

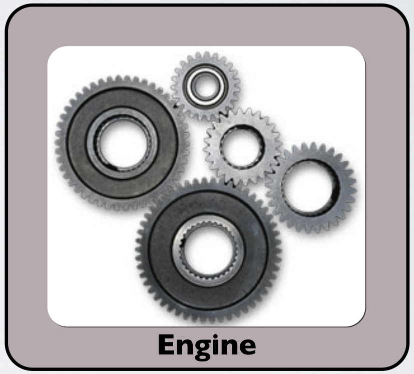
Composer



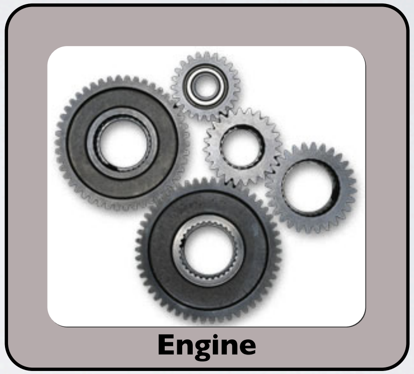
Engine

SCOLA IN PRACTICE

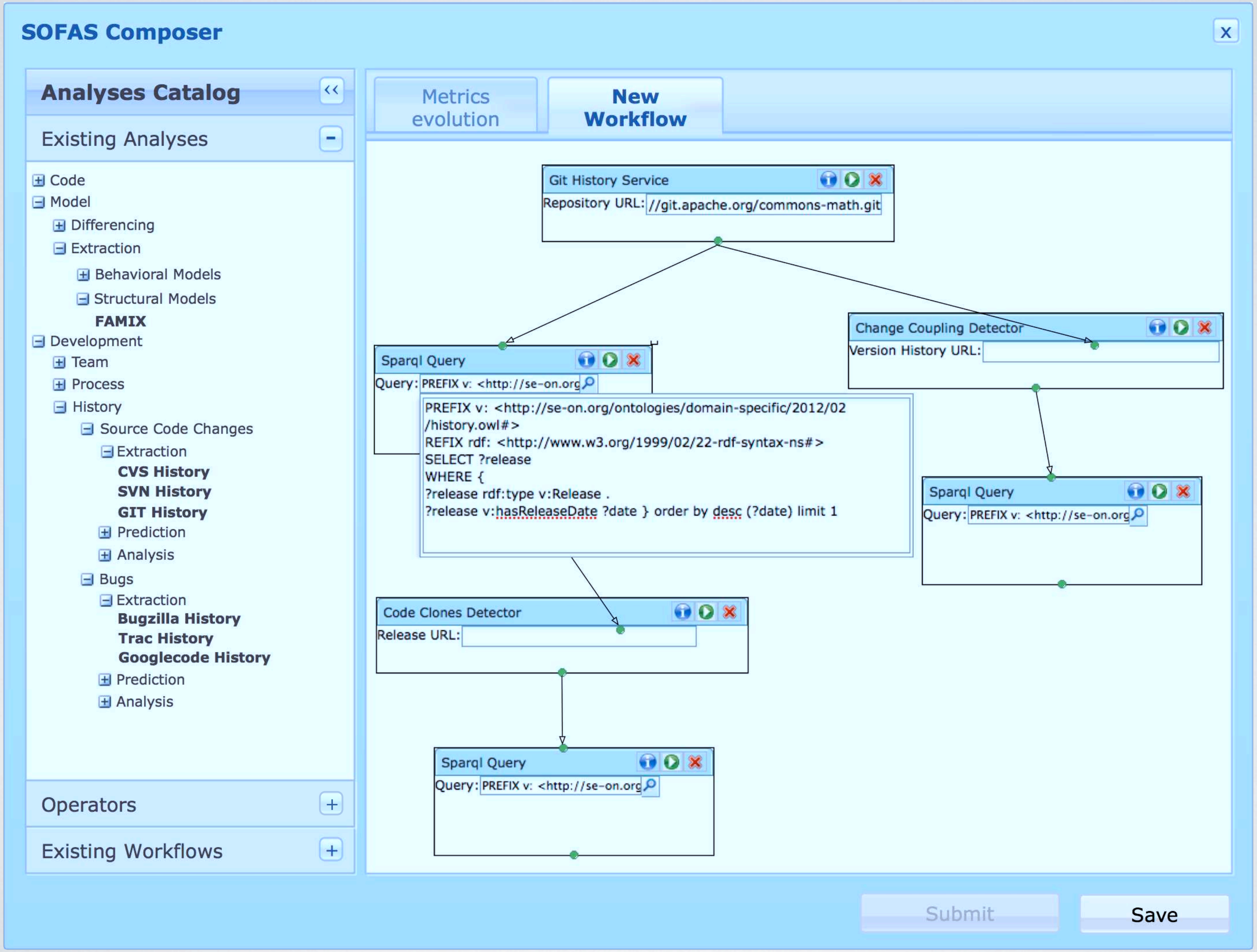




SCOLA IN PRACTICE



SCOLA FOR HUMANS



Repository URL: //git.apache.org/commons-math.git

Version History URL:

Query: PREFIX v: <http://se-on.org

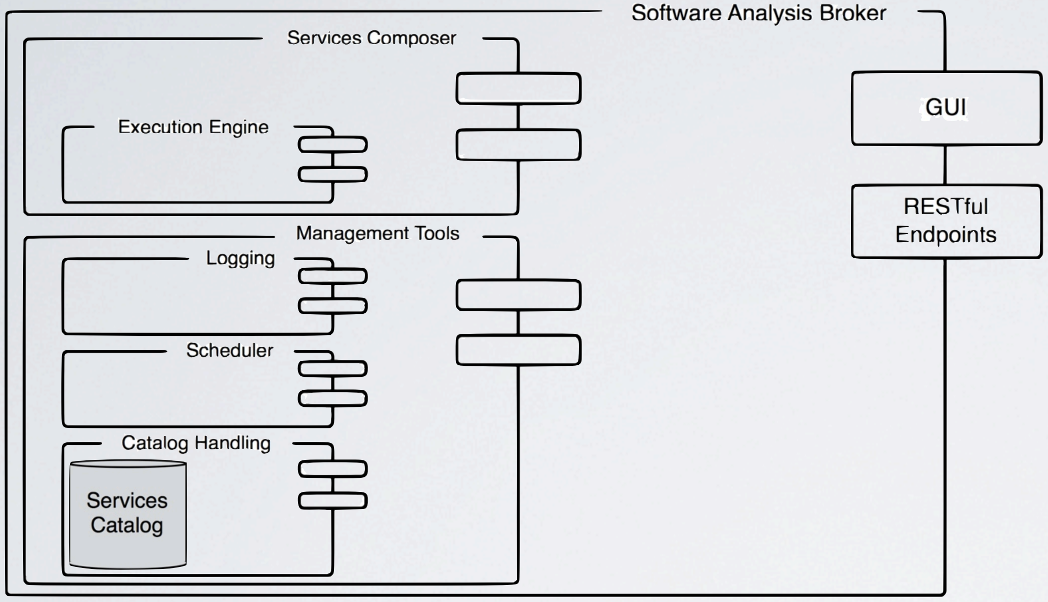
```
PREFIX v: <http://se-on.org/ontologies/domain-specific/2012/02
/history.owl#>
REFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?release
WHERE {
?release rdf:type v:Release .
?release v:hasReleaseDate ?date } order by desc (?date) limit 1
```

Query: PREFIX v: <http://se-on.org

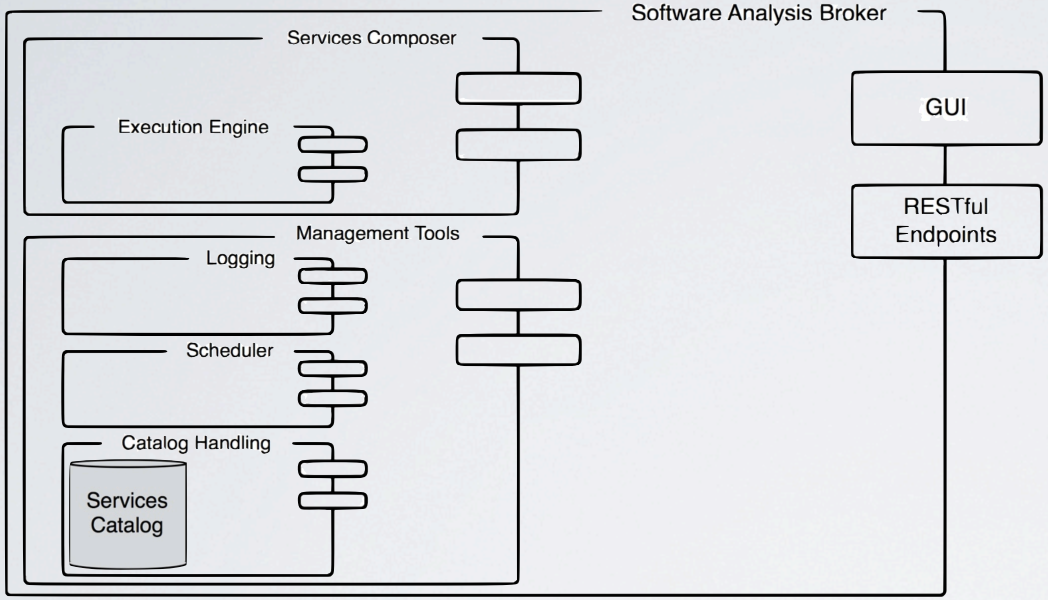
Release URL:

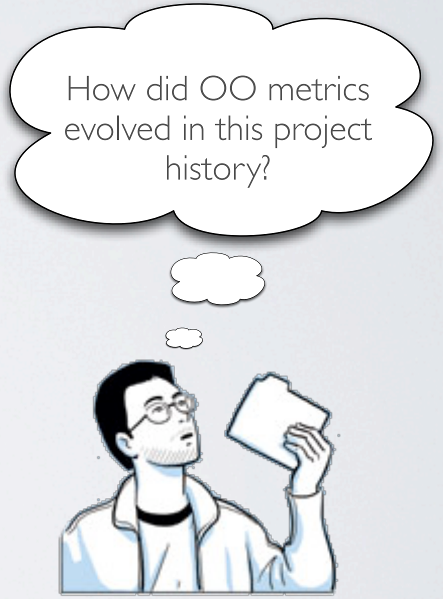
Query: PREFIX v: <http://se-on.org

COMPOSITION IN A NUTSHELL

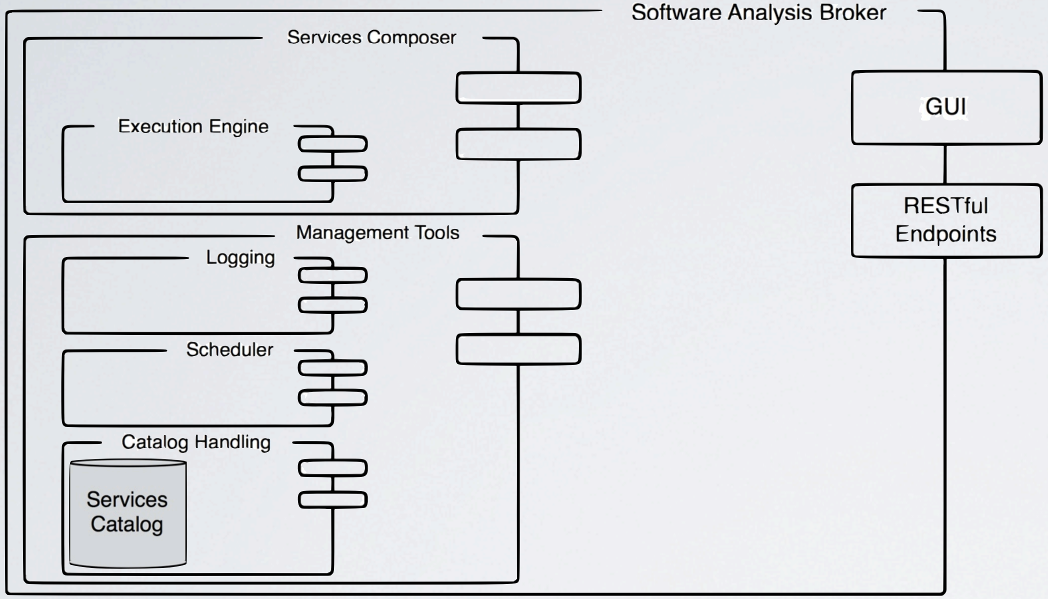


COMPOSITION IN A NUTSHELL



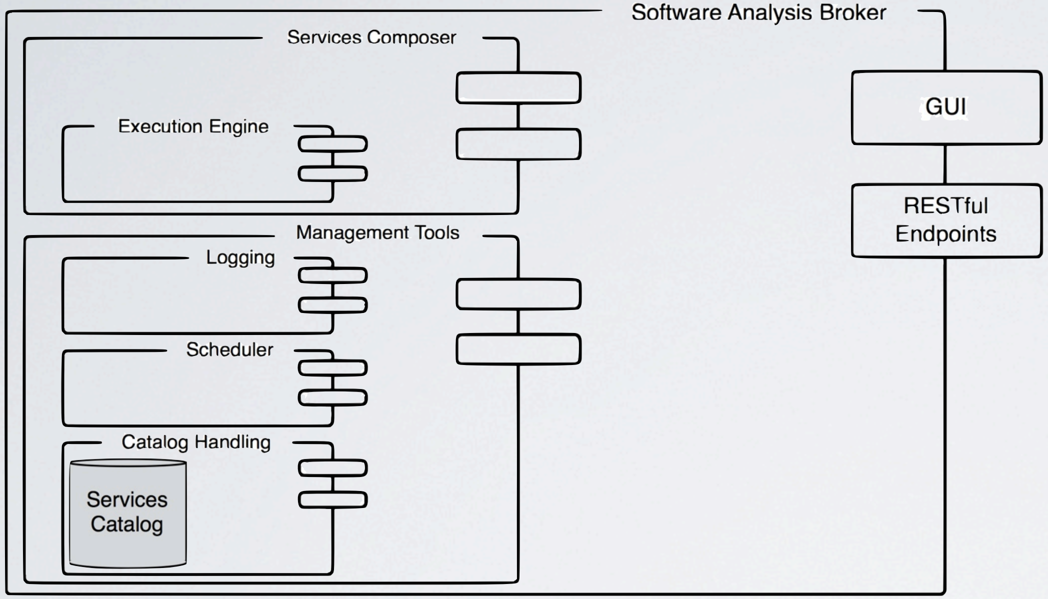


COMPOSITION IN A NUTSHELL



Ok, so let's find a service extracting OO metrics
Famic Model Service
OO Metrics Service

COMPOSITION IN A NUTSHELL


Software Analysis Broker

Services Composer

Execution Engine

Management Tools

Logging

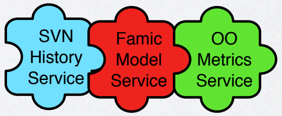
Scheduler

Catalog Handling

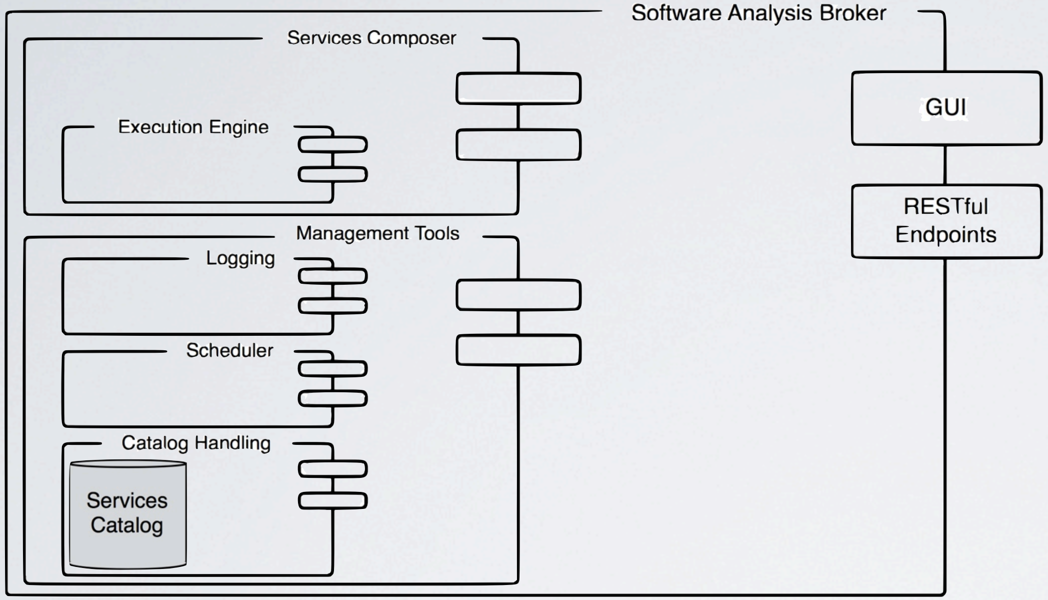
Services Catalog

GUI

RESTful Endpoints



COMPOSITION IN A NUTSHELL



Software Analysis Broker

Services Composer

Execution Engine

Management Tools

Logging

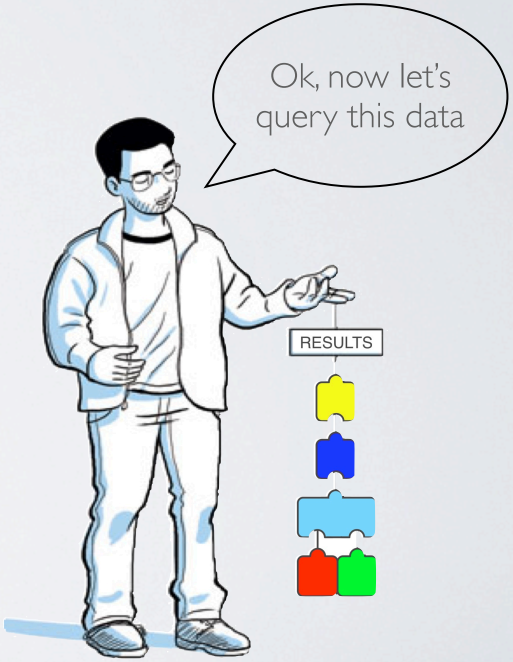
Scheduler

Catalog Handling

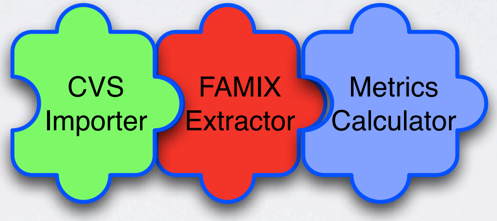
Services Catalog

GUI

RESTful Endpoints



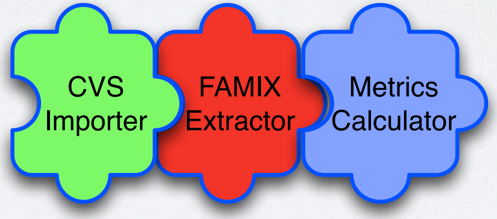
AUTOMATIC COMPOSITION



AUTOMATIC COMPOSITION

```
. . . . . . . . . . . . . . . . . . . . . . . .
<wsdl:types>
<schema elementFormDefault="qualified" targetNamespace="http://ifi.uzh.ch/spec/wsdl/cvs#">
<element name="ExtractionRequest">
<complexType>
<sequence>
<element name="repositoryUrl" type="xsd:string"/>
</sequence>
</complexType>
</element>
<element name="ExtractionResponse">
<complexType>
<sequence>
<element name="projectCVSHistory" type="xsd:string"
sawsdl:modelReference="http://ifi.uzh.ch/ontologies/versioning#VersioningHistory/”>
</sequence>
</complexType>
</element>
</wsdl:types>
. . . . . . . . . . . . . . . . . . . . . . . .
<wsdl:interface name="seal_CVS_Extractor" pattern="http://www.w3.org/ns/wsdl/in-out">
<wsdl:operation name="extractProjectCVS"
sawsdl:modelReference="http://ifi.uzh.ch/ontologies/analysis_taxonomy#CVS_Extraction”>
<wsdl:input element="ExtractionRequest" />
<wsdl:output element="ExtractionResponse" />
</wsdl:operation>
</wsdl:interface>
</wsdl:description>
```

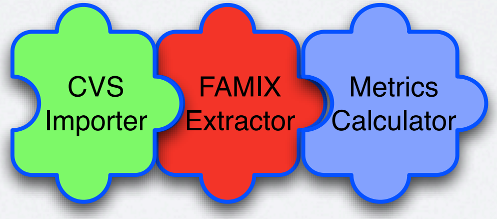
```
. . . . . . . . . . . . . . . . . . . . . . . .
<wsdl:types>
<schema elementFormDefault="qualified" targetNamespace="http://ifi.uzh.ch/spec/wsdl/famix#">
<element name="ExtractionRequest">
<complexType>
<sequence>
<element name="projectCVSHistory" type="xsd:string"
sawsdl:modelReference="http://ifi.uzh.ch/ontologies/versioning#VersioningHistory/”/>
</sequence>
</complexType>
</element>
<element name="ExtractionResponse">
<complexType>
<sequence>
<element name="projectFamixModel" type="xsd:string" maxOccurs="unbounded"
sawsdl:modelReference="http://ifi.uzh.ch/ontologies/famix#FamixModel”/>
<element name="CVS_Famix_Links" type="xsd:string"
sawsdl:modelReference="http://ifi.uzh.ch/ontologies/cvs_famix#CVS_Famix_Links”/>
</sequence>
</sequence>
</complexType>
</element>
</wsdl:types>
. . . . . . . . . . . . . . . . . . . . . . . .
<wsdl:interface name="seal_Famix_Model_Extractor" pattern="http://www.w3.org/ns/wsdl/in-out">
<wsdl:operation name="extractProjectFamix"
sawsdl:modelReference="http://ifi.uzh.ch/ontologies/
analysis_taxonomy#FAMIX_Extraction”>
<wsdl:input element="ExtractionRequest" />
<wsdl:output element="ExtractionResponse" />
</wsdl:operation>
</wsdl:interface>
</wsdl:description>
```



AUTOMATIC COMPOSITION

```
. . . . . . . . . . . . . . . . . . . . . . . .
<wsdl:types>
<schema elementFormDefault="qualified" targetNamespace="http://ifi.uzh.ch/spec/wsdl/cvs#">
<element name="ExtractionRequest">
<complexType>
<sequence>
<element name="repositoryUrl" type="xsd:string"/>
</sequence>
</complexType>
</element>
<element name="ExtractionResponse">
<complexType>
<sequence>
<element name="projectCVSHistory" type="xsd:string"
sawsdl:modelReference="http://ifi.uzh.ch/ontologies/versioning#VersioningHistory/”>
</sequence>
</complexType>
</element>
</wsdl:types>
. . . . . . . . . . . . . . . . . . . . . . . .
<wsdl:interface name="seal_CVS_Extractor" pattern="http://www.w3.org/ns/wsdl/in-out">
<wsdl:operation name="extractProjectCVS"
sawsdl:modelReference="http://ifi.uzh.ch/ontologies/analysis_taxonomy#CVS_Extraction”>
<wsdl:input element="ExtractionRequest" />
<wsdl:output element="ExtractionResponse" />
</wsdl:operation>
</wsdl:interface>
</wsdl:description>
```

```
. . . . . . . . . . . . . . . . . . . . . . . .
<wsdl:types>
<schema elementFormDefault="qualified" targetNamespace="http://ifi.uzh.ch/spec/wsdl/famix#">
<element name="ExtractionRequest">
<complexType>
<sequence>
<element name="projectCVSHistory" type="xsd:string"
sawsdl:modelReference="http://ifi.uzh.ch/ontologies/versioning#VersioningHistory/”/>
</sequence>
</complexType>
</element>
<element name="ExtractionResponse">
<complexType>
<sequence>
<element name="projectFamixModel" type="xsd:string" maxOccurs="unbounded"
sawsdl:modelReference="http://ifi.uzh.ch/ontologies/famix#FamixModel”/>
<element name="CVS_Famix_Links" type="xsd:string"
sawsdl:modelReference="http://ifi.uzh.ch/ontologies/cvs_famix#CVS_Famix_Links”/>
</sequence>
</sequence>
</complexType>
</element>
</wsdl:types>
. . . . . . . . . . . . . . . . . . . . . . . .
<wsdl:interface name="seal_Famix_Model_Extractor" pattern="http://www.w3.org/ns/wsdl/in-out">
<wsdl:operation name="extractProjectFamix"
sawsdl:modelReference="http://ifi.uzh.ch/ontologies/
analysis_taxonomy#FAMIX_Extraction”>
<wsdl:input element="ExtractionRequest" />
<wsdl:output element="ExtractionResponse" />
</wsdl:operation>
</wsdl:interface>
</wsdl:description>
```



What is the impact of the proposed approach?

EVALUATION

- Focused on the 3 major milestones, assessing 3 main properties:
    1. Applicability
    2. Flexibility
    3. Relevance
- Mainly based on use cases

EVALUATION

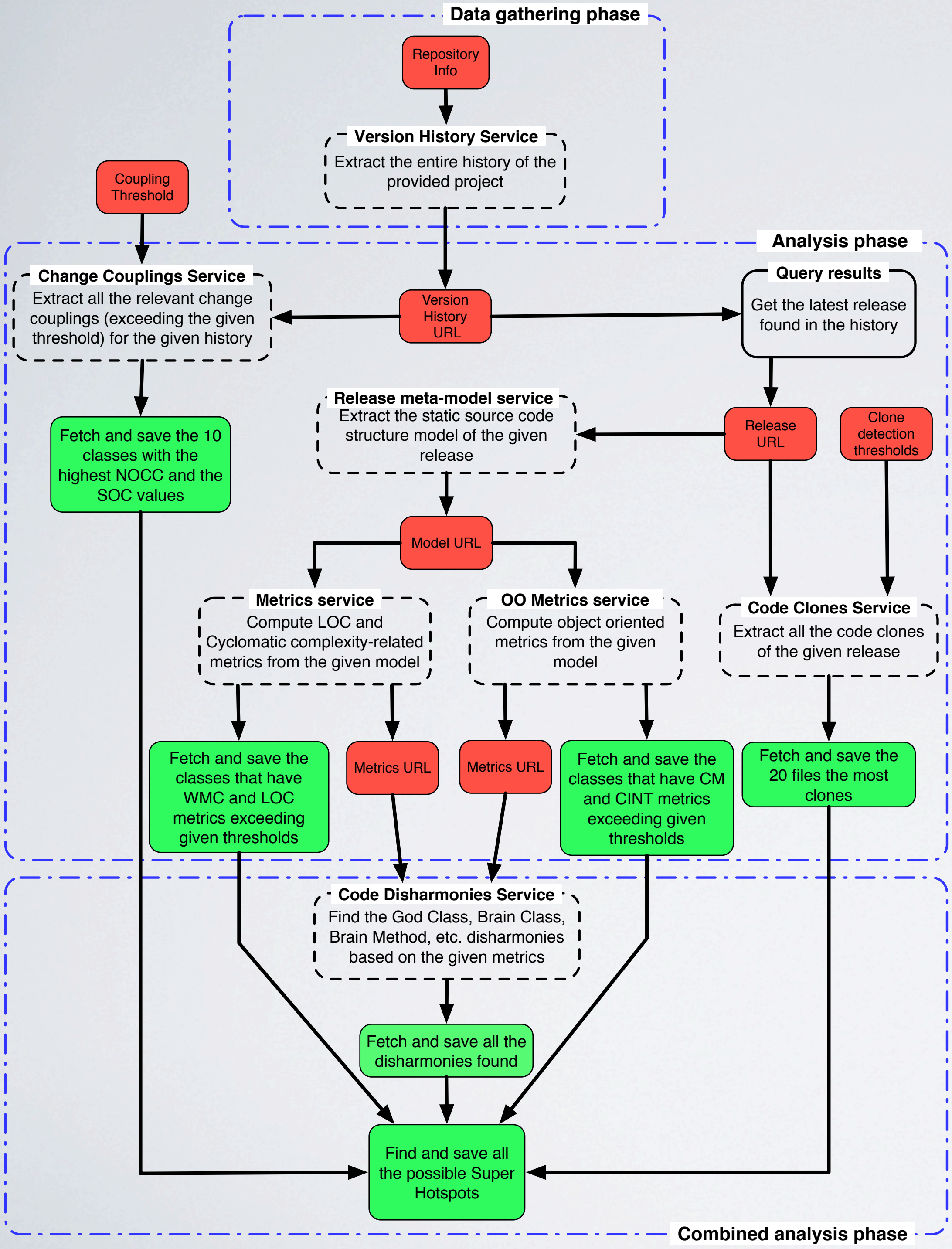
- Two use cases based on the quality audit of a commercial, mission critical system (~400k LOC):

1. We conceptually prove that our framework can be used to address relevant evolution analysis questions

   ➡We answer the question "*Which are the hotspots and evolution anomalies for a project?*"

2. We demonstrate how tools can harness SCoLa to automatically gather a wide range of varied yet interlinked information about a software system and use it

   ➡**Software Evolution Facets**



Data gathering phase
Repository Info
Version History Service
Extract the entire history of the provided project
Coupling Threshold
Analysis phase
Change Couplings Service
Extract all the relevant change couplings (exceeding the given threshold) for the given history
Version History URL
Query results
Get the latest release found in the history
Release meta-model service
Extract the static source code structure model of the given release
Release URL
Clone detection thresholds
Fetch and save the 10 classes with the highest NOCC and the SOC values
Model URL
Metrics service
Compute LOC and Cyclomatic complexity-related metrics from the given model
OO Metrics service
Compute object oriented metrics from the given model
Code Clones Service
Extract all the code clones of the given release
Fetch and save the classes that have WMC and LOC metrics exceeding given thresholds
Metrics URL
Metrics URL
Fetch and save the classes that have CM and CINT metrics exceeding given thresholds
Fetch and save the 20 files the most clones
Code Disharmonies Service
Find the God Class, Brain Class, Brain Method, etc. disharmonies based on the given metrics
Fetch and save all the disharmonies found
Find and save all the possible Super Hotspots
Combined analysis phase

- “Which are the hotspots and evolution anomalies for a project?”
    - Which entities have high or abnormal values of known code quality metrics?
    - Which entities present specific code smells?
    - Which entities have a high change coupling?
    - Which entities have a lot of copied code (code clones)?
    - Which entities are super hotspots?

DEMO

PERFORMANCE

- **Apache HTTPD** history ~2h:15m (~1g of vca data)
- **Commercial system** study (~400K LOC) ~3hrs
  - history: ~30min
  - meta-model extraction of 12 releases: 2 hrs
  - metrics calculation for every selected release: 30min
  - code disharmonies calculation for every selected release:10 min

SYSTEMS STUDIED

- The history of the entire Apache foundation codebase (336 projects)
- The metrics history of some of the most popular Java OSS projects
  - **Azureus, ArgoUML, Eclipse**
- Some of the most popular OSS projects (based on OHLOH stats)
  - **Mozilla, Linux kernel, PostgreSQL, Python, Ruby,** etc.

RELATED WORK

- Tool-based systems: **Kenyon**, **Evolizer**, **softChange**
  - ad-hoc developed tools and techniques
  - no composition and usage of different, independent analyses
  - no combination into custom combinations
- **OASIS**
  - allow an analysis available in one tool to use the fact-base of another one
  - domain ontology to describe a set of representational concepts that the different tools to be integrated require and support
- Data repositories
  - **Flossmole** => more high-level development information and dynamics and is offered "as-is"
  - **Ultimate Debian DB** => system specific information

RELATED WORK

- Online analysis platforms
  - **FOSSology**
    - framework to analyze source code with different, custom analyses that can be created by users to fulfill their specific needs
    - currently it only features an agent that detects the code license
  - **Alitheia Core**
    - plugin-based
    - still highly centralized approach => similar to Kenyon or Evolizer but online accessible

SOFAS’ OFFSPRING

- Interest in solutions enabling the sharing and/or combination of analyses and their results both from researchers and the industry
- Plugin-based VCS
- SOFAS 2.0, a.k.a applied SOFAS
- Software Evolution Facets

A Plugin-based Version Control System for Software (Evolution) Analysis

Motivation (the good)

- VCSes greatly evolved since their introduction in the 70s
- Their core functionality and rationale has stayed the same (check in, check out, commit, view the history...)
- They are good for forward engineering activities
- However, there is much to be learnt from the history they contain (MSR teaches)

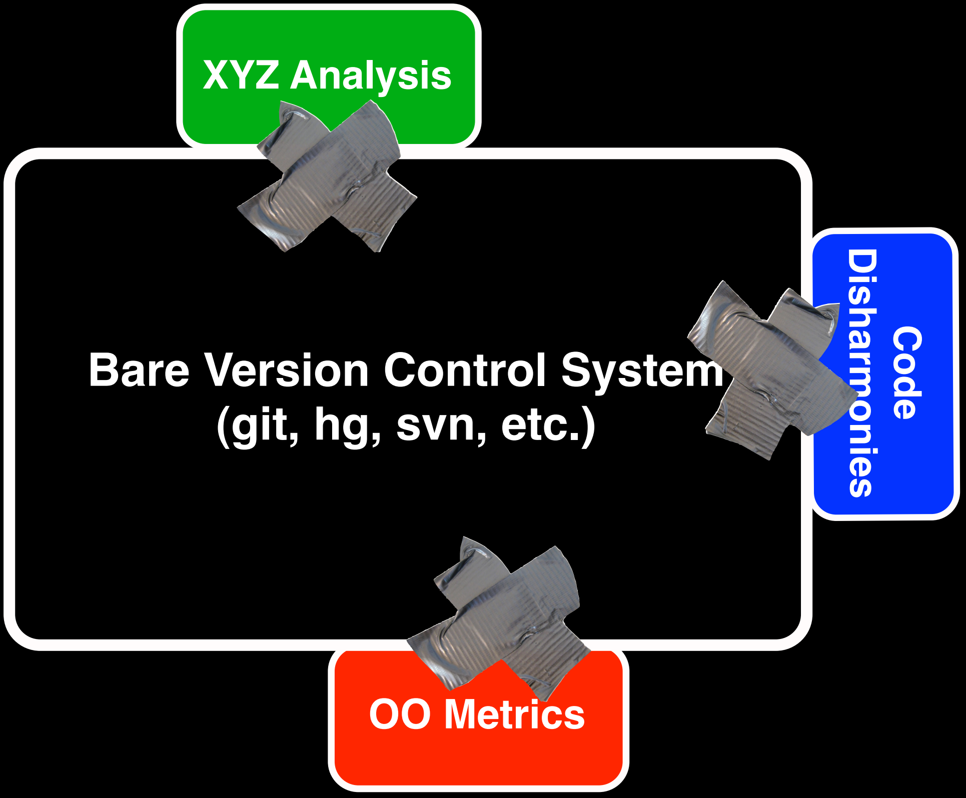
Motivation (the bad)

- Version Control systems are NOT meant to be systematically analyzed
- They still are "backup" systems
- Analyses interface with them through the bare history log
- The software miners rushed for the "gold" but forgot to improve the working conditions in the mines

The Current Situation

Bare Version Control System
(git, hg, svn, etc.)

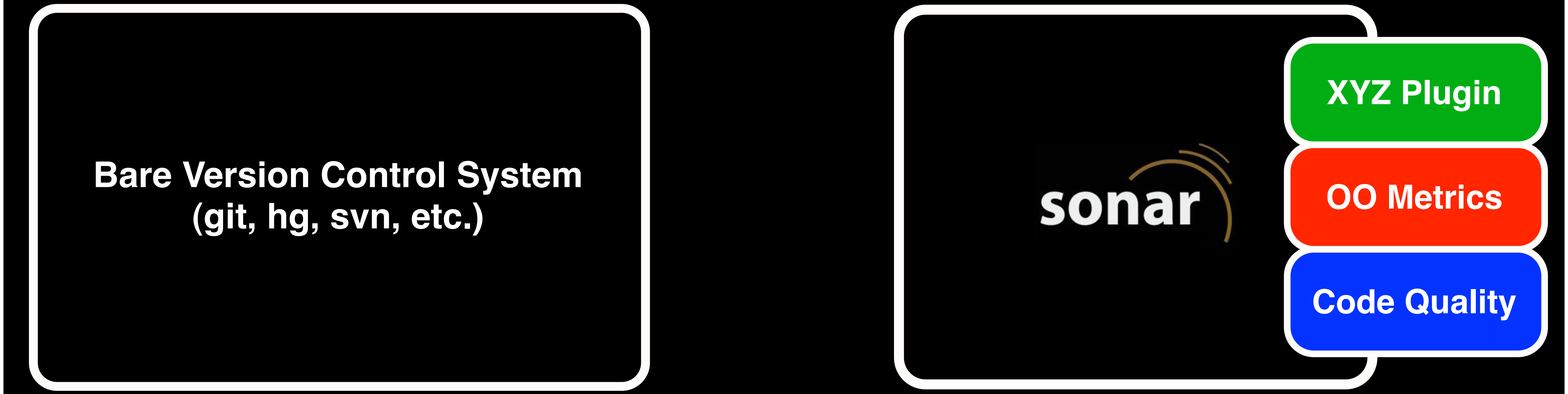
The Current Situation



An Existing Solution

Bare Version Control System
(git, hg, svn, etc.)

An Existing Solution



Bare Version Control System
(git, hg, svn, etc.)

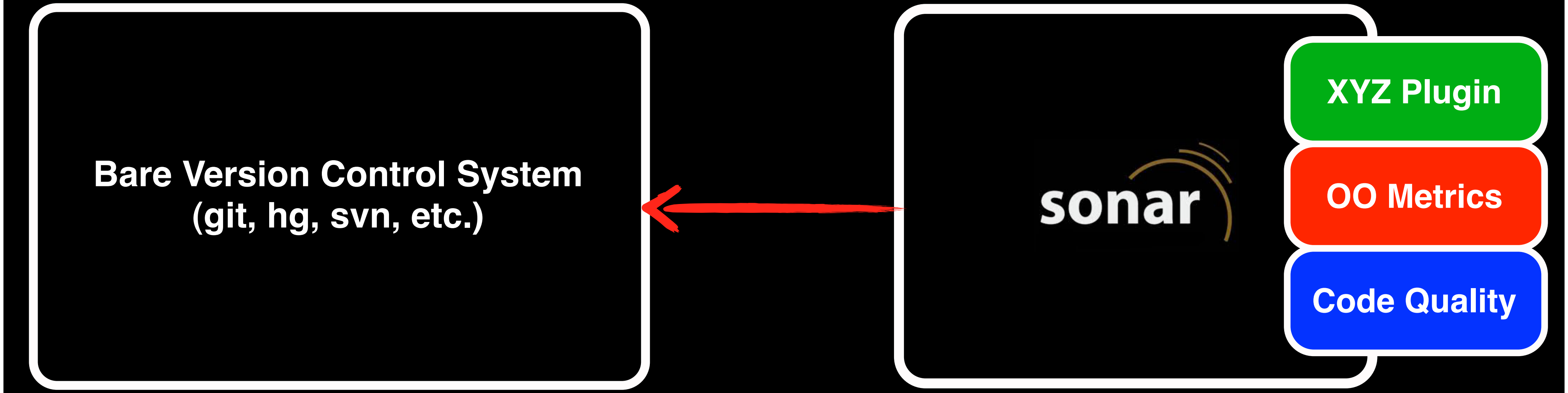
sonar

XYZ Plugin

OO Metrics

Code Quality

An Existing Solution



Bare Version Control System (git, hg, svn, etc.)

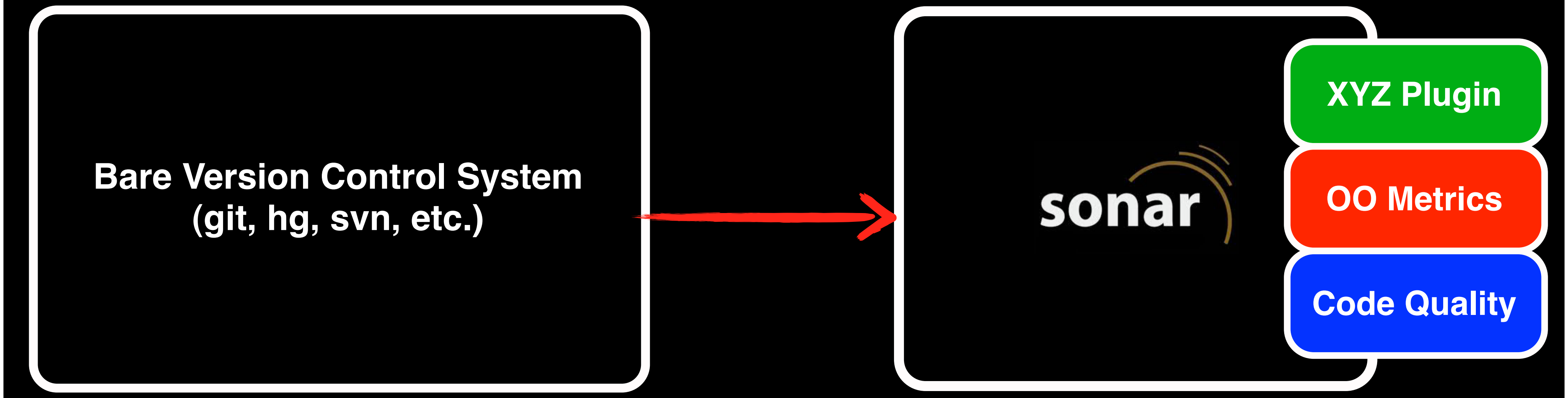
sonar

XYZ Plugin

OO Metrics

Code Quality

An Existing Solution



Bare Version Control System
(git, hg, svn, etc.)

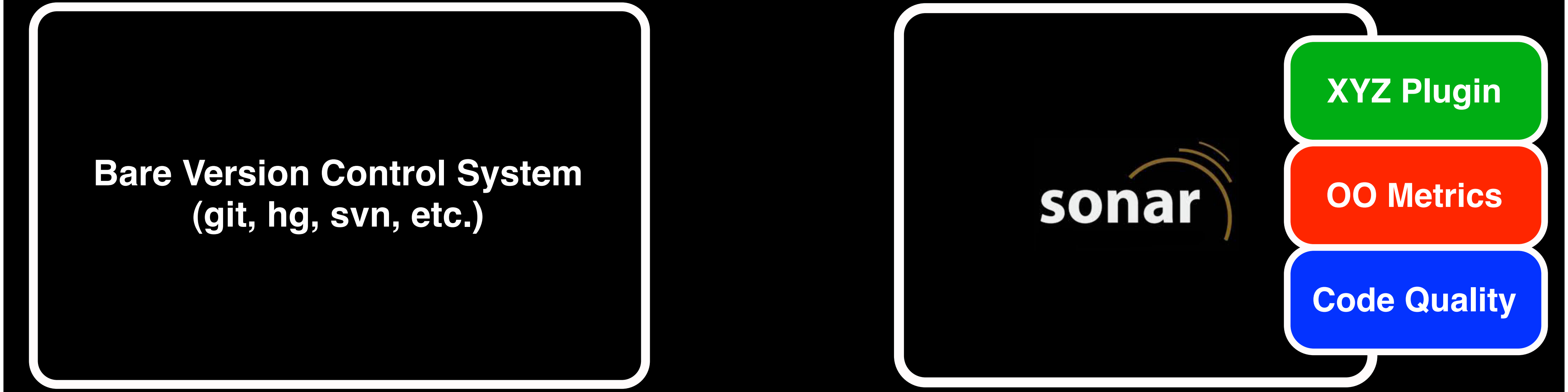
sonar

XYZ Plugin

OO Metrics

Code Quality

An Existing Solution



Bare Version Control System
(git, hg, svn, etc.)

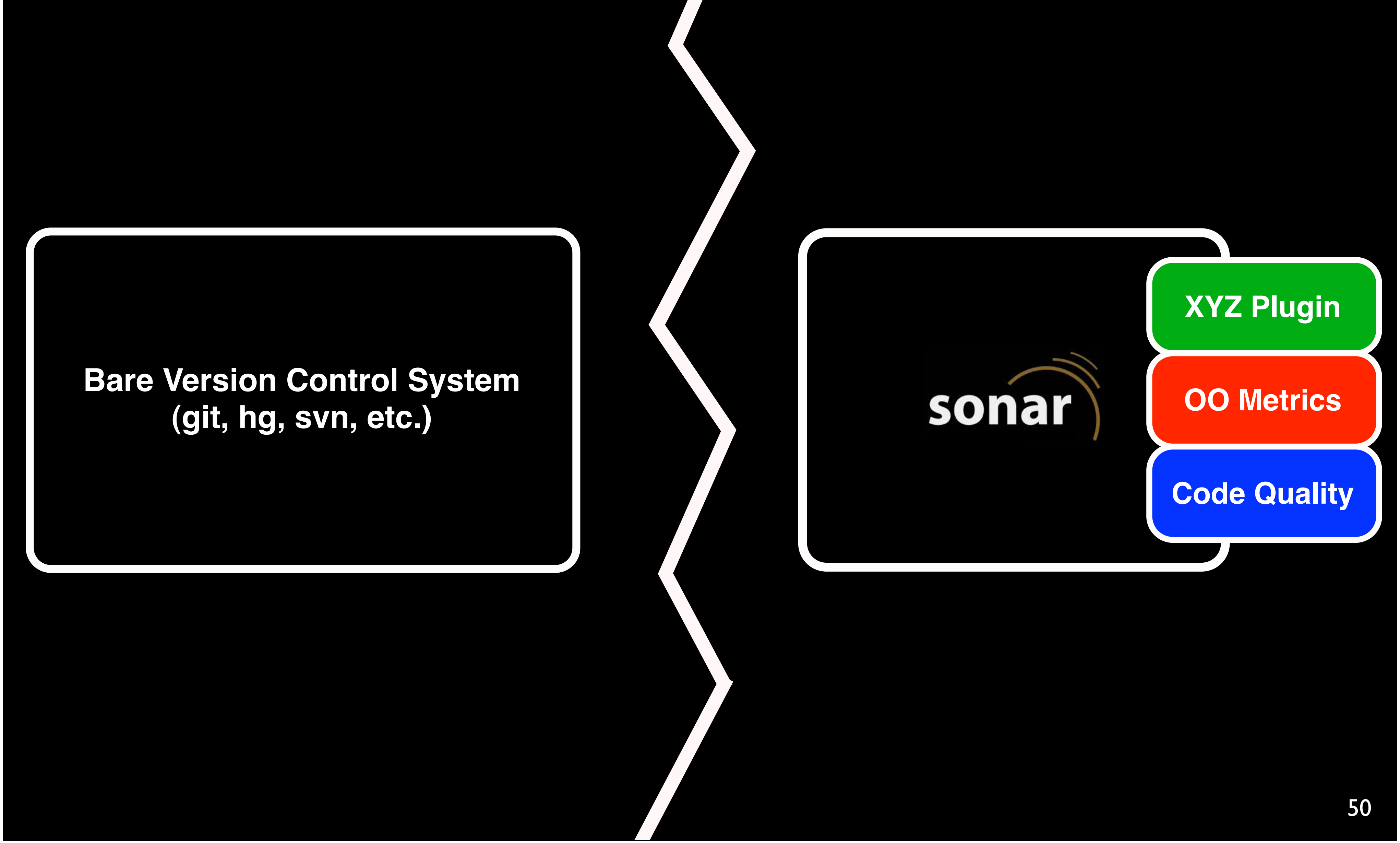
sonar

XYZ Plugin

OO Metrics

Code Quality

An Existing Solution



Bare Version Control System
(git, hg, svn, etc.)

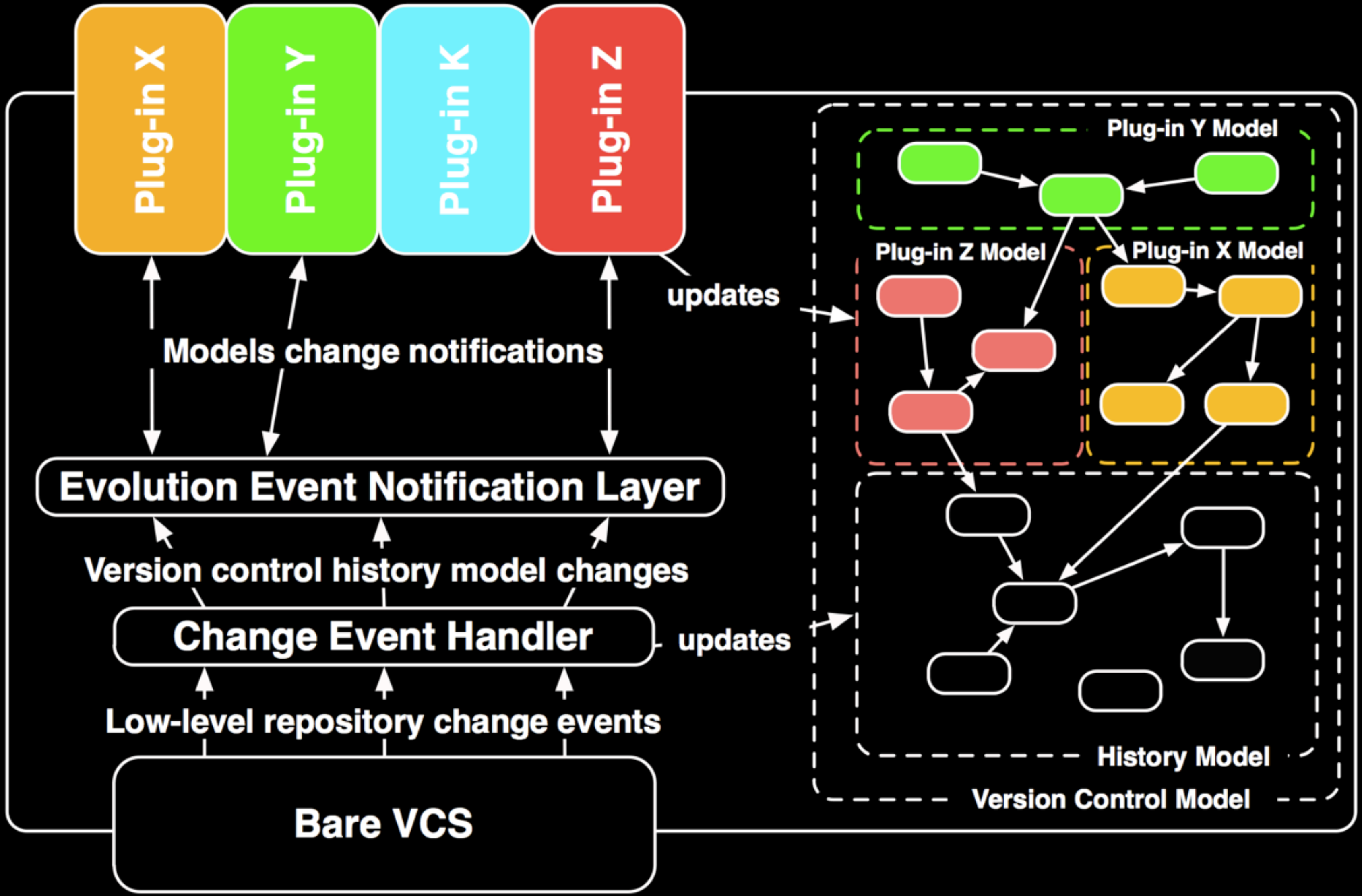
sonar

XYZ Plugin

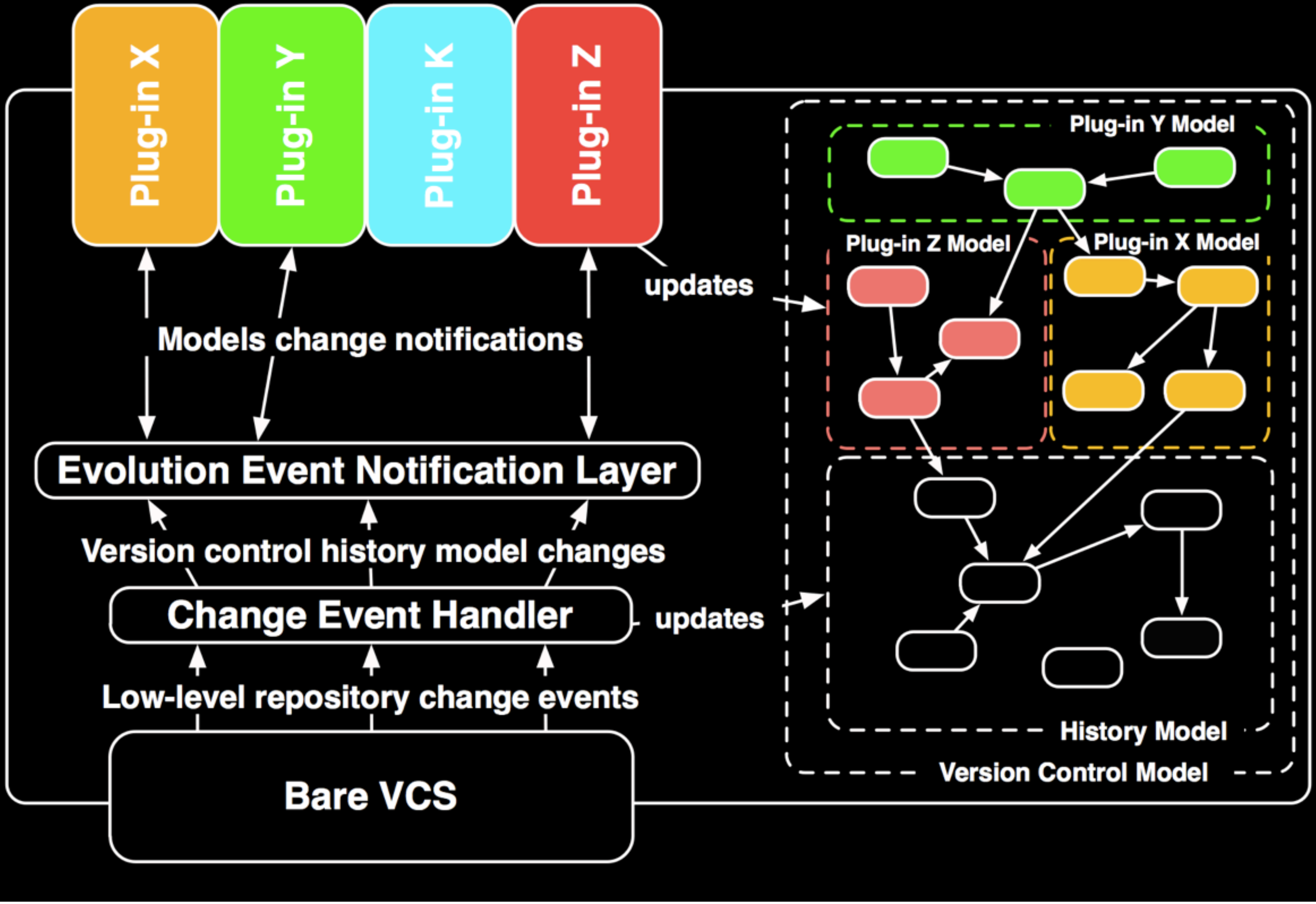
OO Metrics

Code Quality

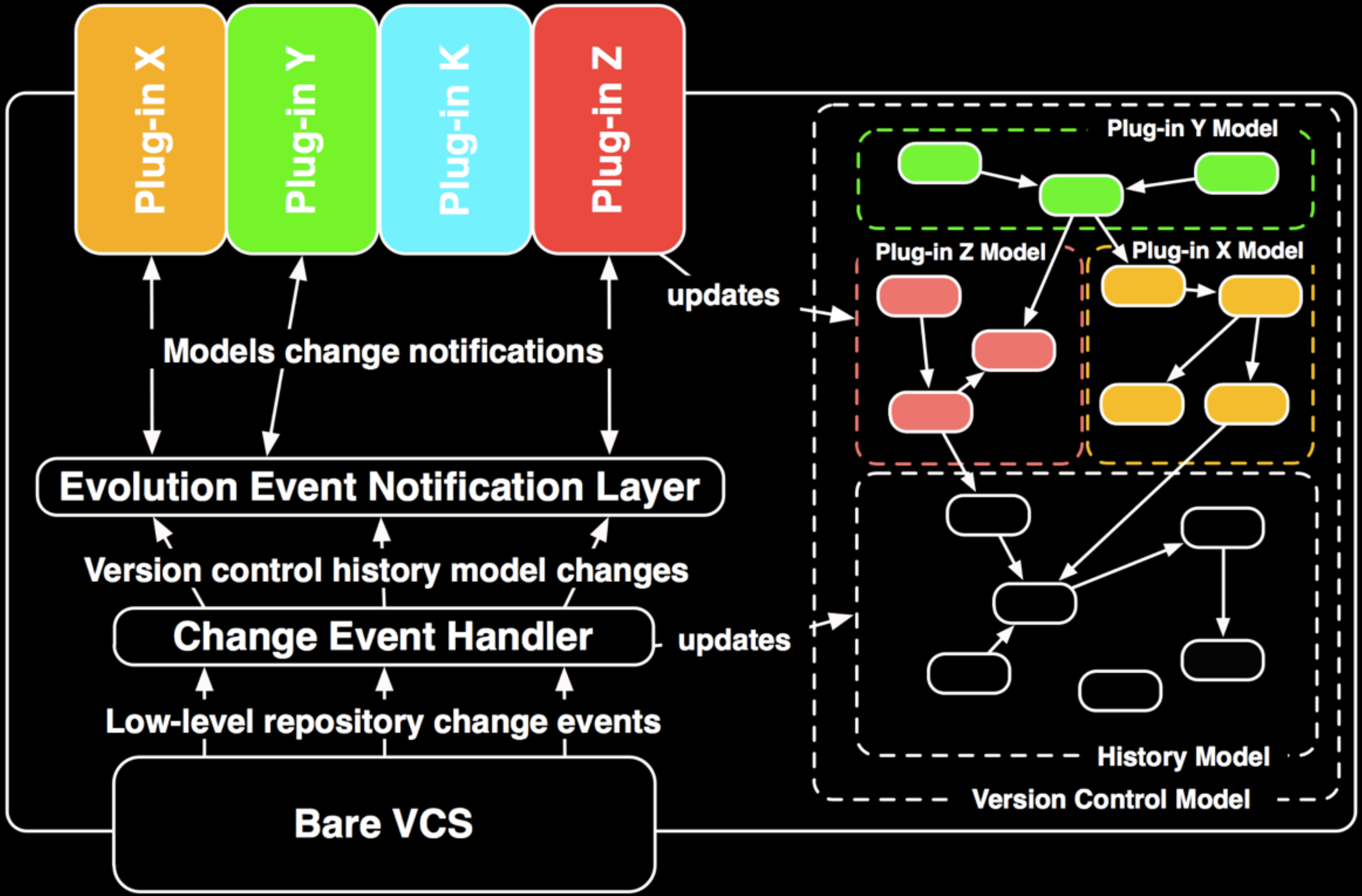
Architecture



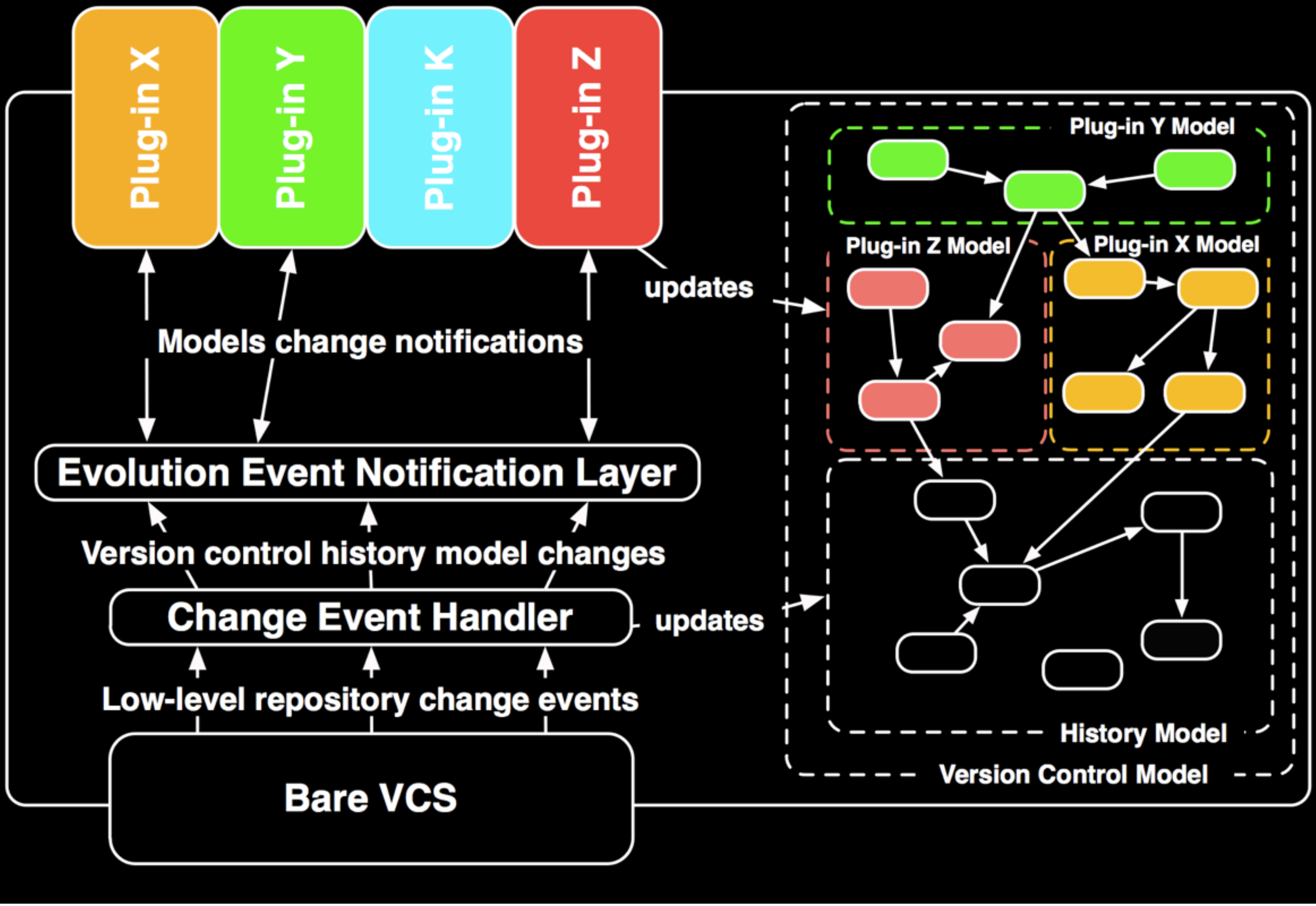
Change Event Handler



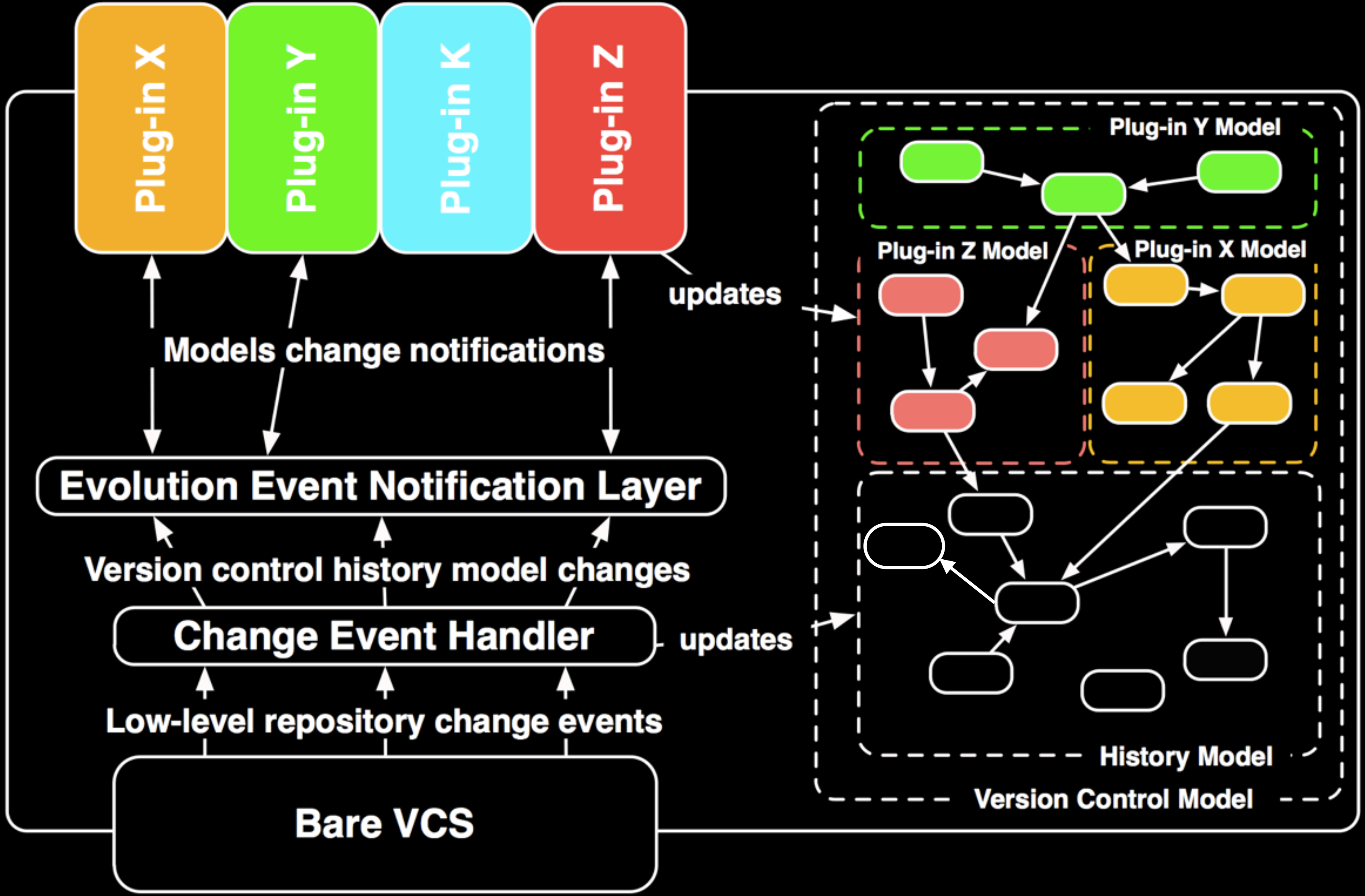
Change Event Handler



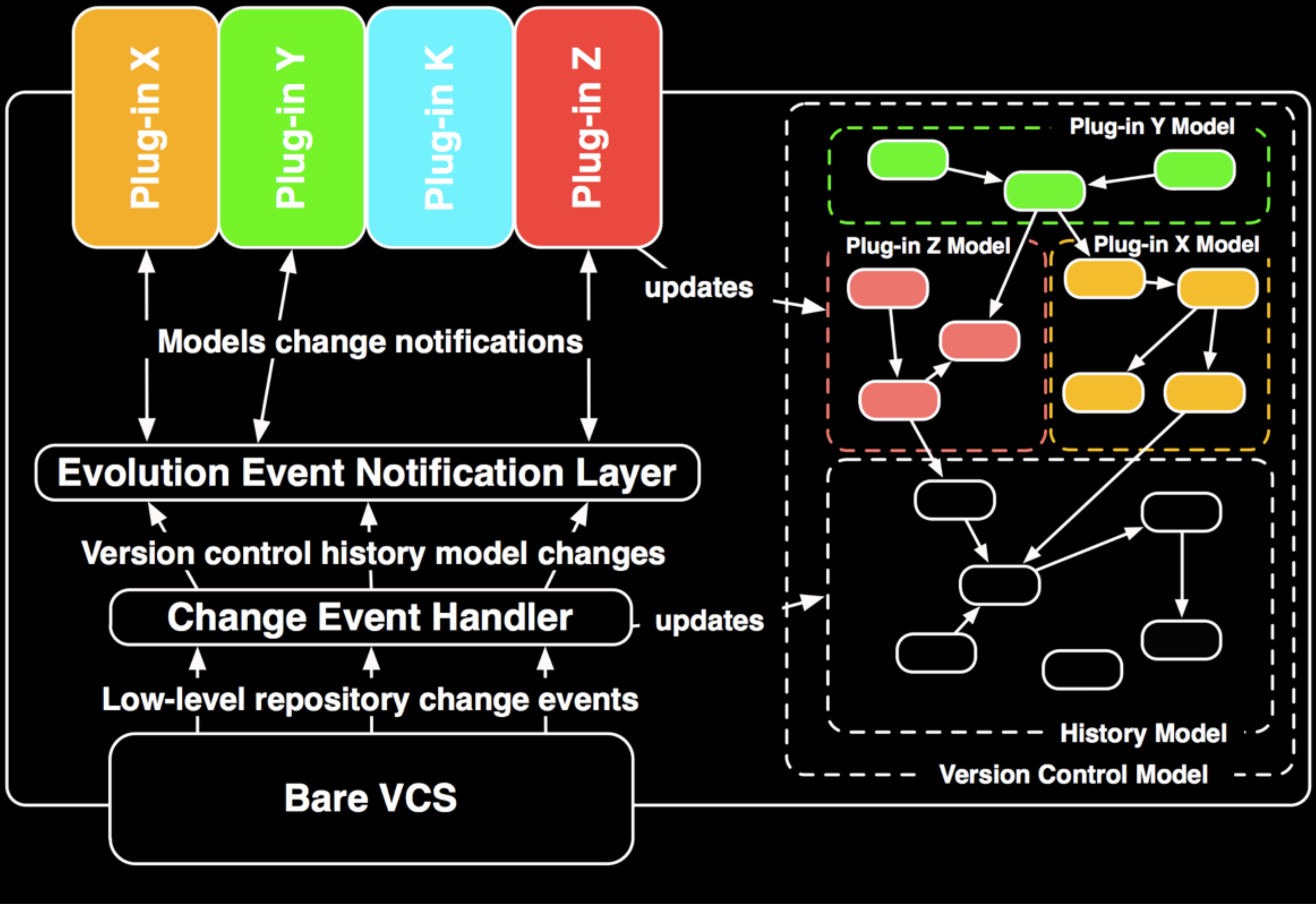
Change Event Handler



Change Event Handler

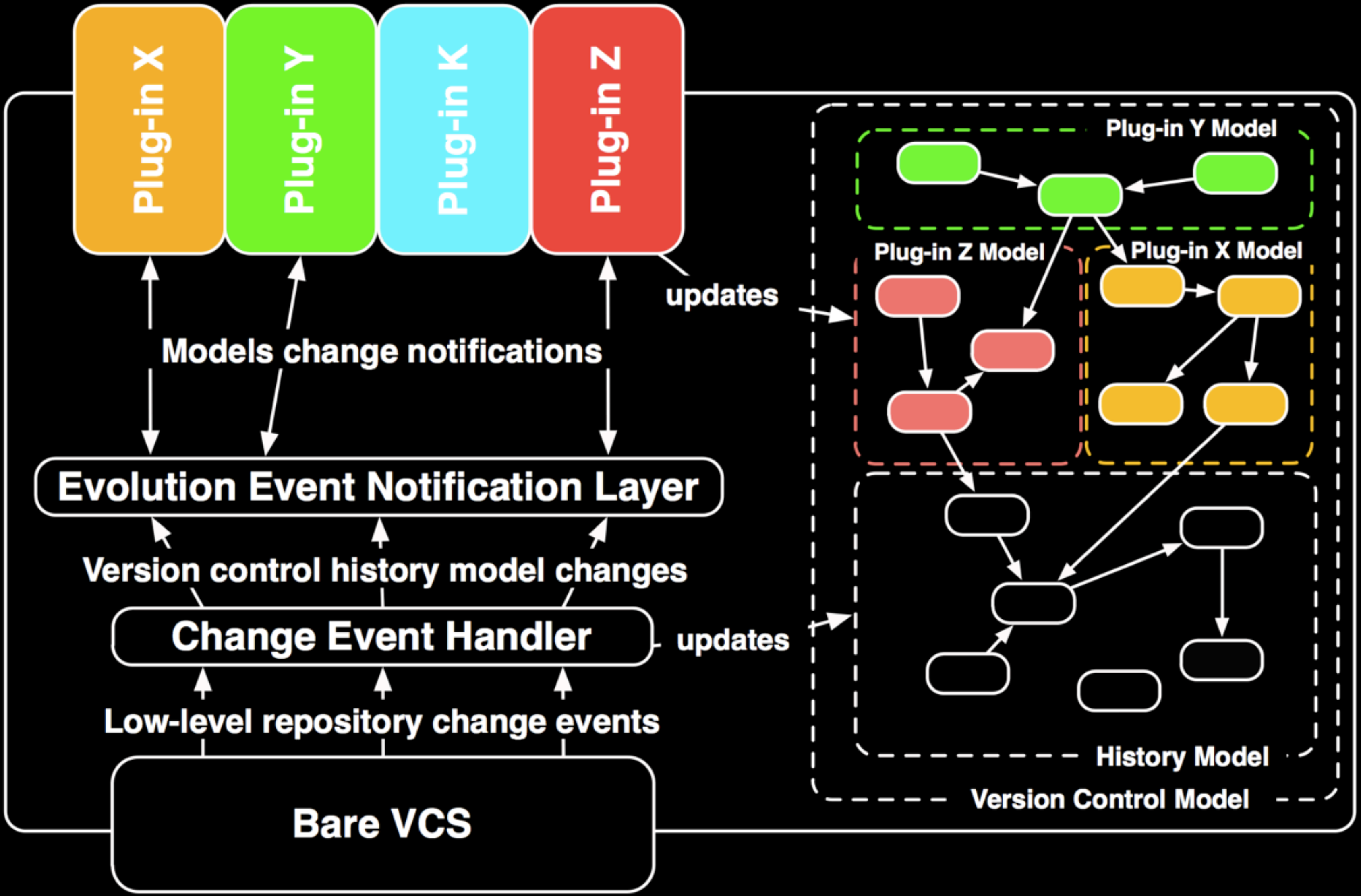


Version Control History Model

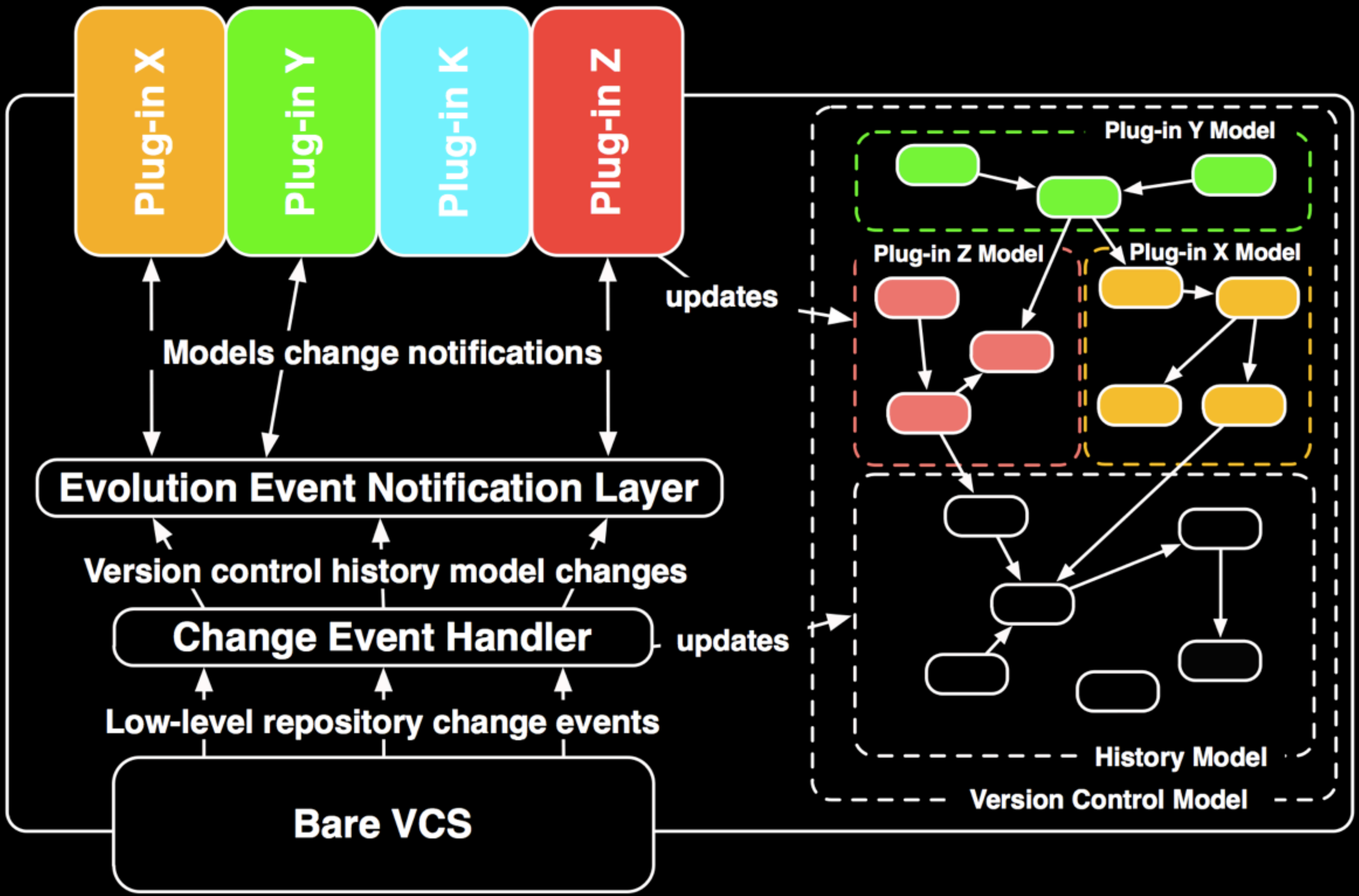


Version Control History Model

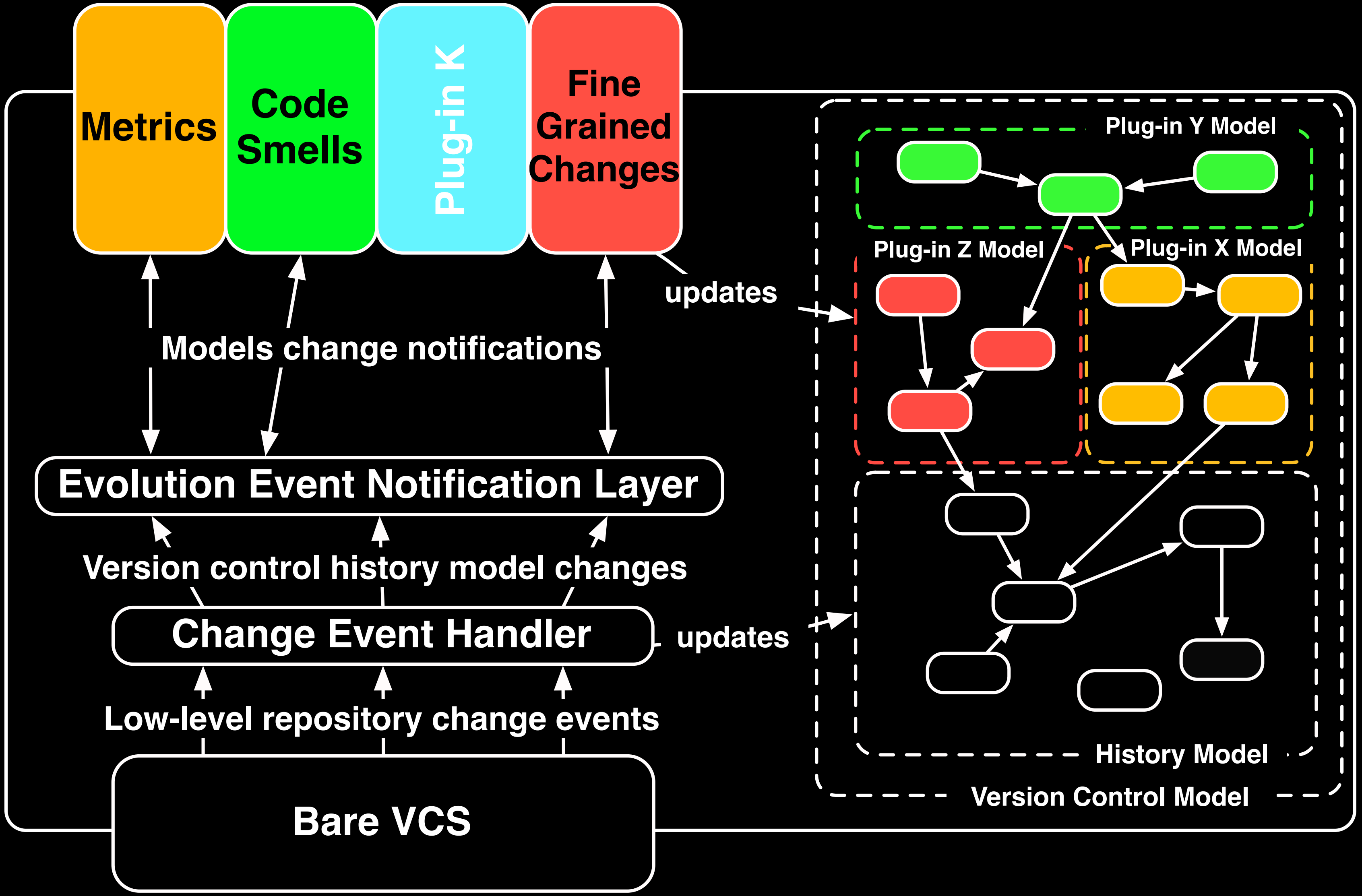
Event Notification Layer



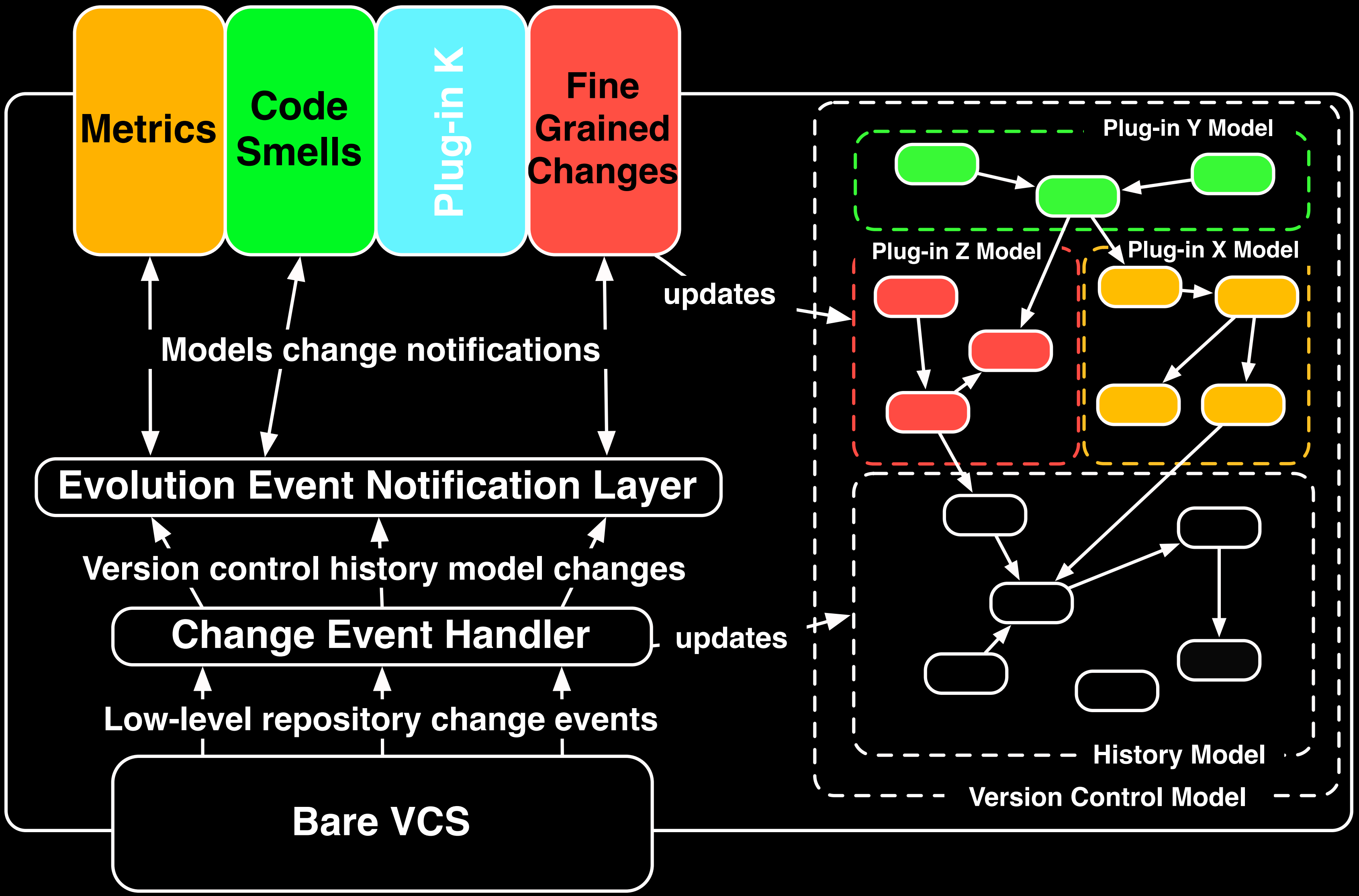
Event Notification Layer



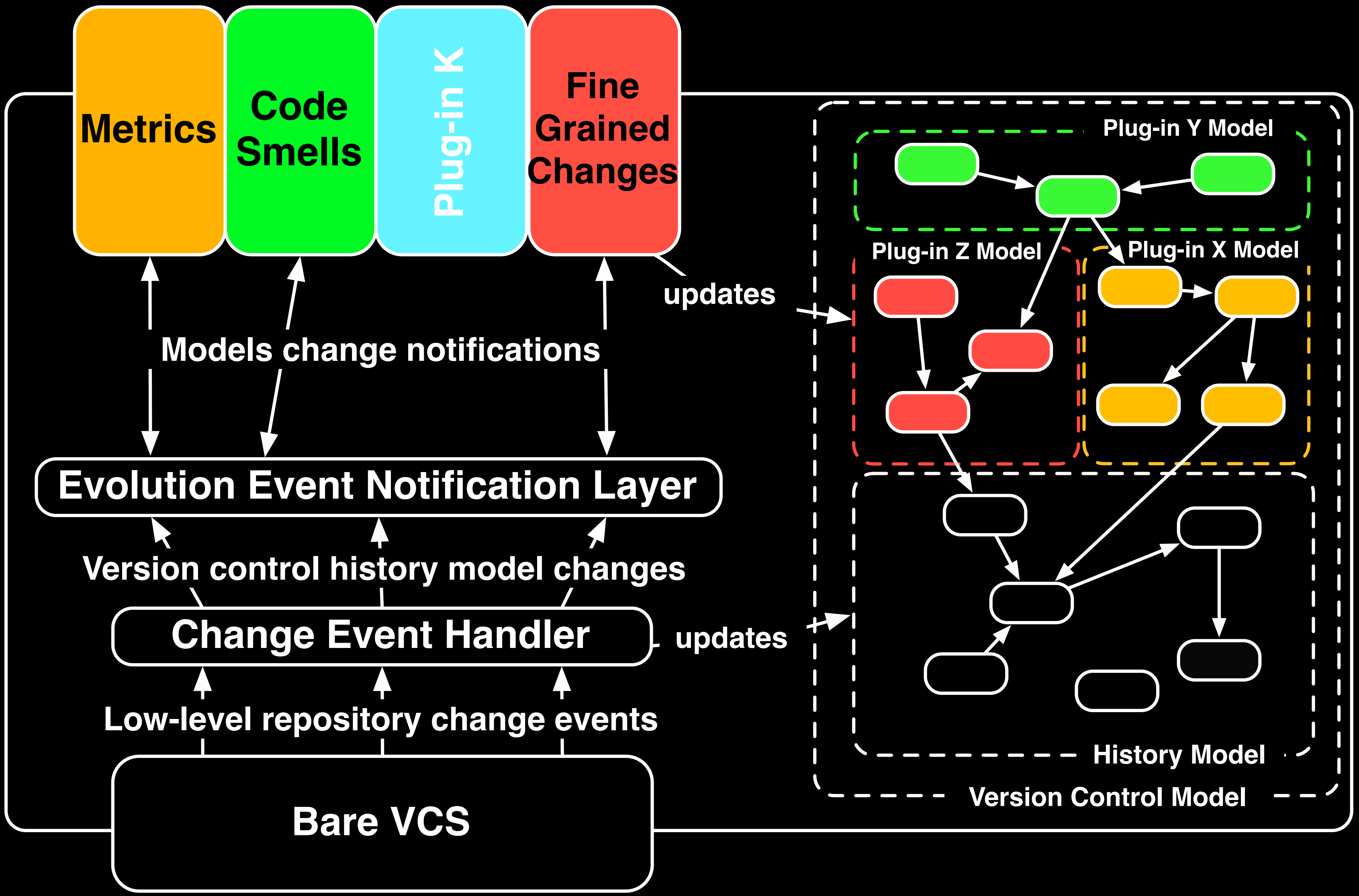
Plug-Ins


Metrics
Code Smells
Plug-in K
Fine Grained Changes
Models change notifications
updates
Evolution Event Notification Layer
Version control history model changes
Change Event Handler
updates
Low-level repository change events
Bare VCS
Plug-in Y Model
Plug-in Z Model
Plug-in X Model
History Model
Version Control Model

Plug-Ins


Metrics
Code Smells
Plug-in K
Fine Grained Changes
updates
Models change notifications
Evolution Event Notification Layer
Version control history model changes
Change Event Handler
updates
Low-level repository change events
Bare VCS
Plug-in Y Model
Plug-in Z Model
Plug-in X Model
History Model
Version Control Model

Plug-Ins


Metrics
Code Smells
Plug-in K
Fine Grained Changes
updates
Models change notifications
Evolution Event Notification Layer
Version control history model changes
Change Event Handler
updates
Low-level repository change events
Bare VCS
Plug-in Y Model
Plug-in Z Model
Plug-in X Model
History Model
Version Control Model

Usage Scenario 1

- Continuous code quality
  - Source code metrics are calculated as soon as a change is committed
  - Enriches the version control history model with metrics data
  - Additional plugins can exploit such data, e.g. Code Disharmonies calculator

Usage Scenario 2

- Fine grained source code changes
  - VCS keep track of changes in a simplistic way: no change semantic!
  - Existing solution are "a posteriori" and time consuming
  - Change Distiller plugin activated by every commit

Usage Scenario 3

- Querying the historical data from the outside
  - VCSes have UI to navigate the repositories
  - No way to programmatically query them
  - Query interface plugins to query the repository with different languages

Conclusions

- A first proof of concept prototype has been developed with 2 plugins
- A more sound implementation is on the way for a possible first case study
- We do not want to re-invent the wheel but to enhance it
- We want to bridge the practice-software evolution analysis gap