SOFtware Analysis as a Service

NUIs as Looking Glass for Software Architecture

Harald Gall

seal.ifi.uzh.ch

University of Zurich

s.e.a.l.

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

Show me the architecture!

- Architecture + analysis go together!

Microsoft Visual Studio 2010, Architecture Explorer

Theme:
Support Architecting by Software Analysis Services consolidated in NUIs

Bits and Pieces

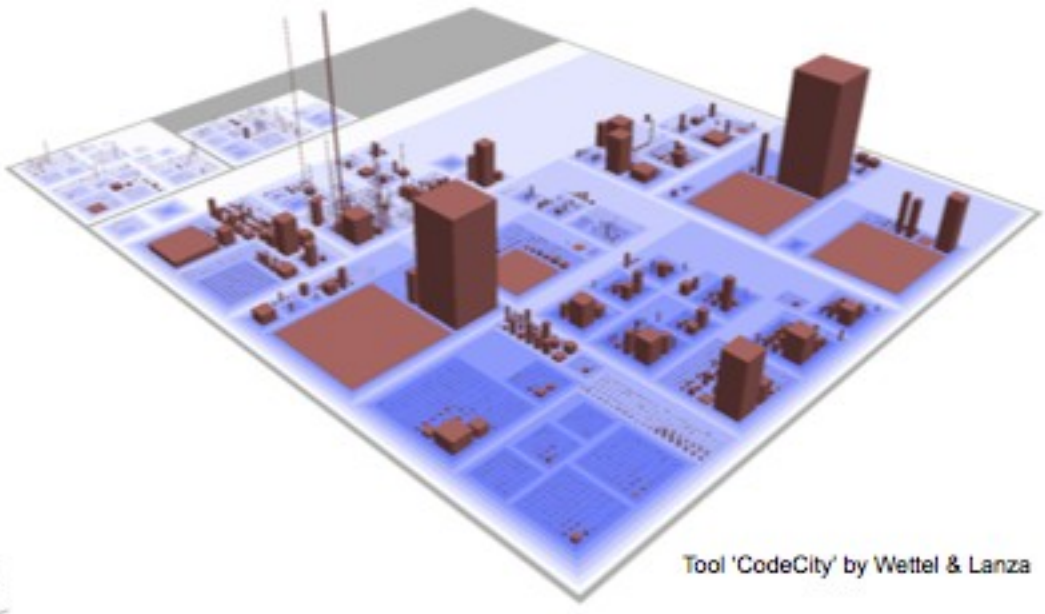
Bits & Pieces

- Software analysis (SA) in context
- SA as Services
- NUIs for SA
- Some Visions

Illustration:
A Metaphor for Software

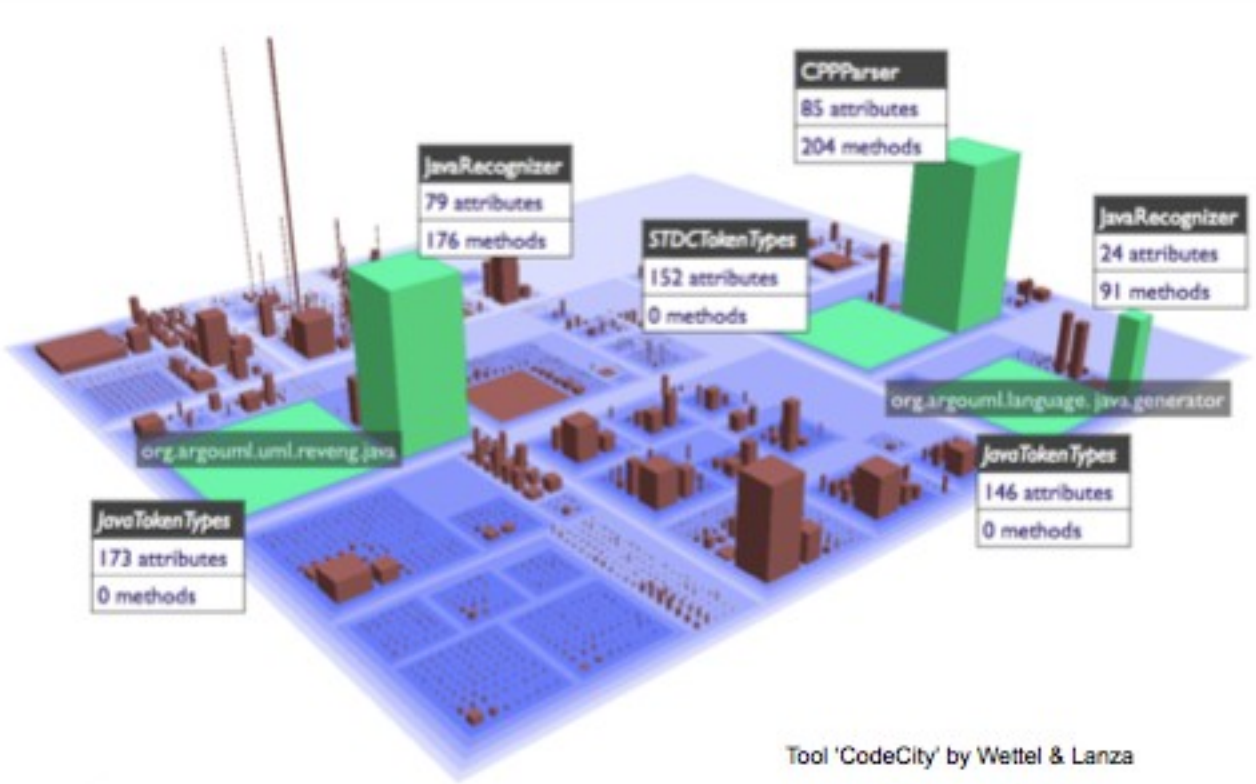
s.e.a.l.

Code City: Software Architecture



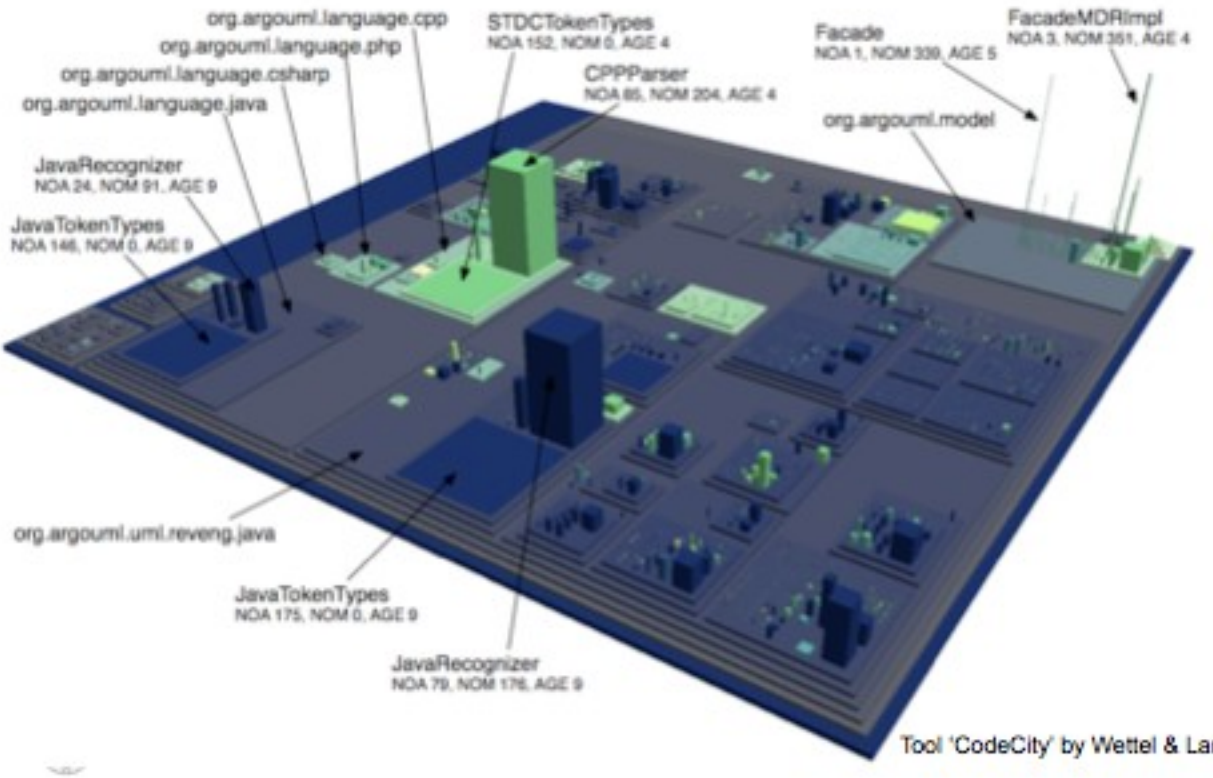
Tool 'CodeCity' by Wettel & Lanza

ArgoUML City



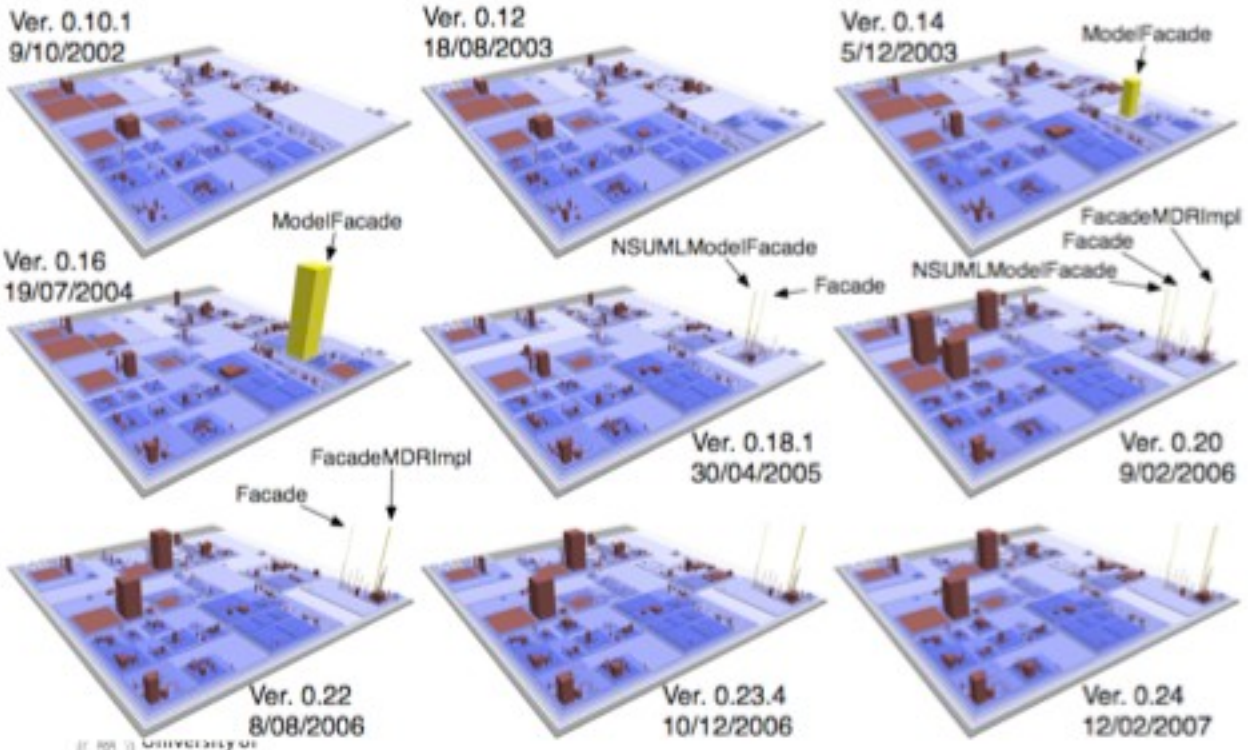
Tool 'CodeCity' by Wettel & Lanza

The age of a City

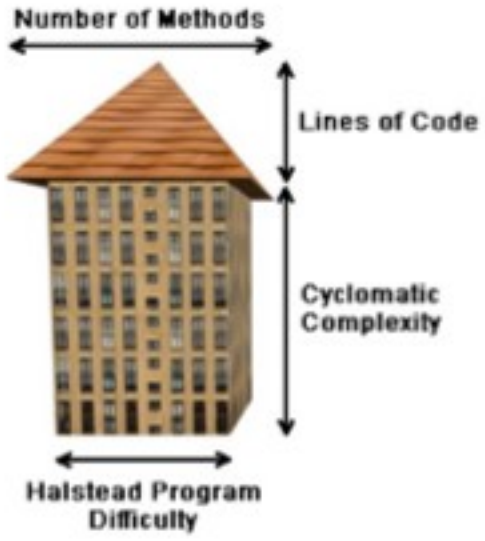


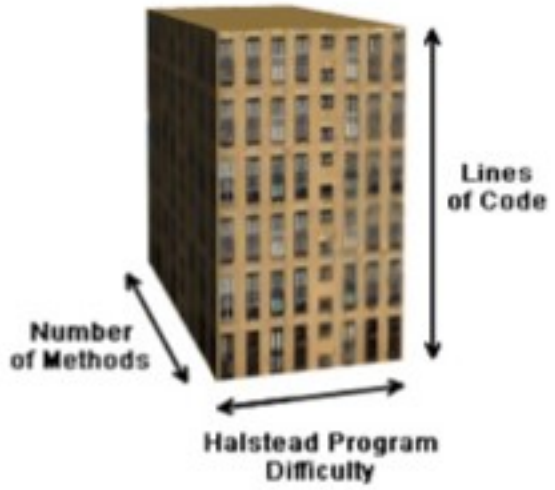
Tool 'CodeCity' by Wettel & Lanza

Evolution of a City

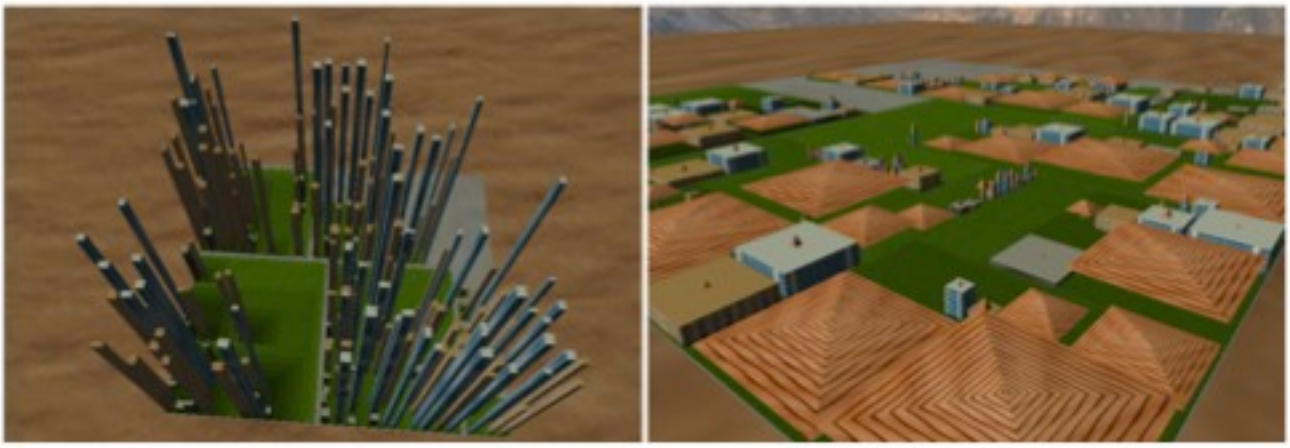


Houses: a closer look





Metric look forms a City



University of Zurich

Software Evolution Analyses

s.e.a.l.

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

What do we want to know?

Does distributed development affect software quality? (Windows Vista)

Cross-project defect prediction: when does it work?

Visual (Effort Estimation) Patterns in Issue Tracking Data

Analyzing the co-evolution of Comments and Source Code

Predicting the fix time of bugs

Supporting Developers with Natural Language Queries

Can Developer-Module Networks Predict Failures?

Interactive Views for Analyzing Problem Reports

A Tool for Visual Understanding of Source Code Dependencies

Where's the Problem?

Analyses are hard to access and use by outsiders

Hardly any replicability apart from some few virtuous exceptions

Cross-application and cross-domain data integration is, at best, cumbersome

People keep reinventing the wheel

Challenges

1. How can these analyses be offered and composed in a consistent way?

2. How can data coming from different analyses be effectively integrated, queried and used?

3. How is this approach different from existing ones? What are its novel benefits?

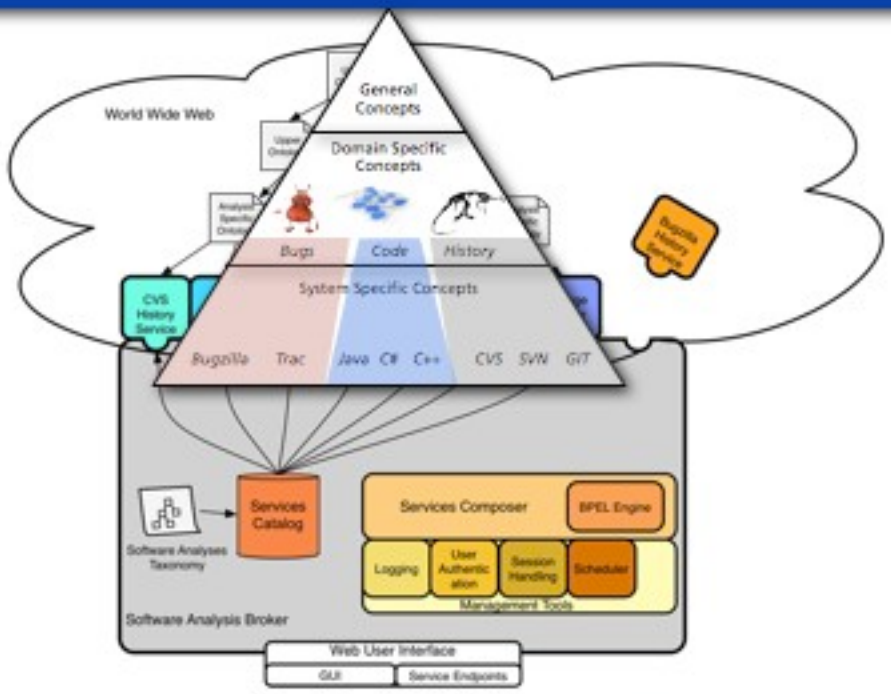
Software Analysis as a Service

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

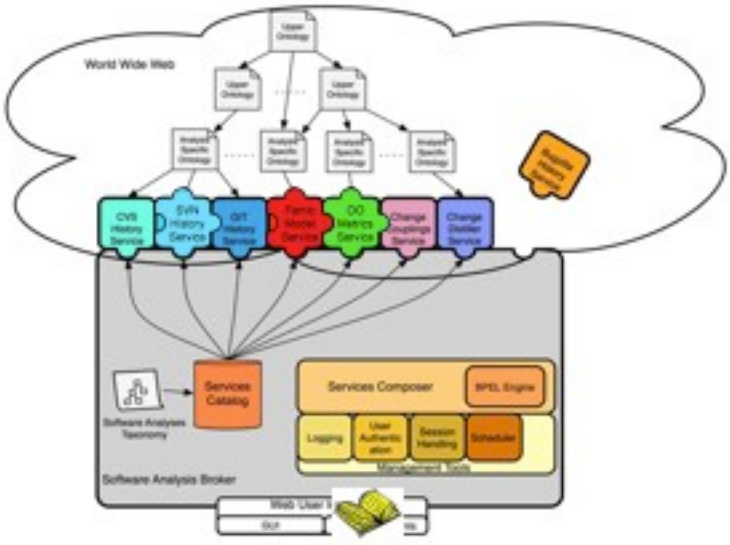
SOFtware Analysis Services

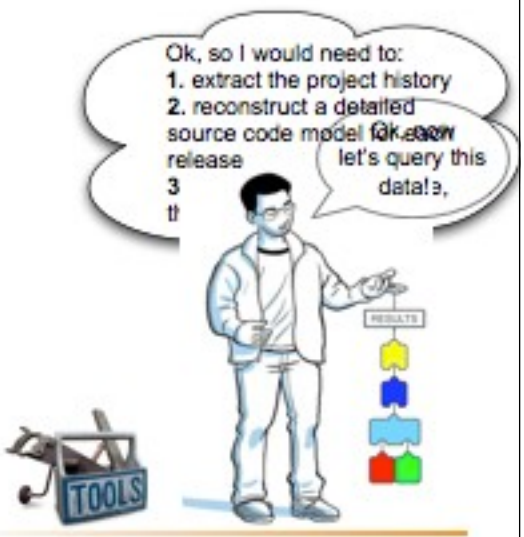
- The actual analysis is offered as a service
- The user just selects the analysis with the data to be analyzed and gets the results
- Data is key; analyses can be lengthy and expensive!

Approach

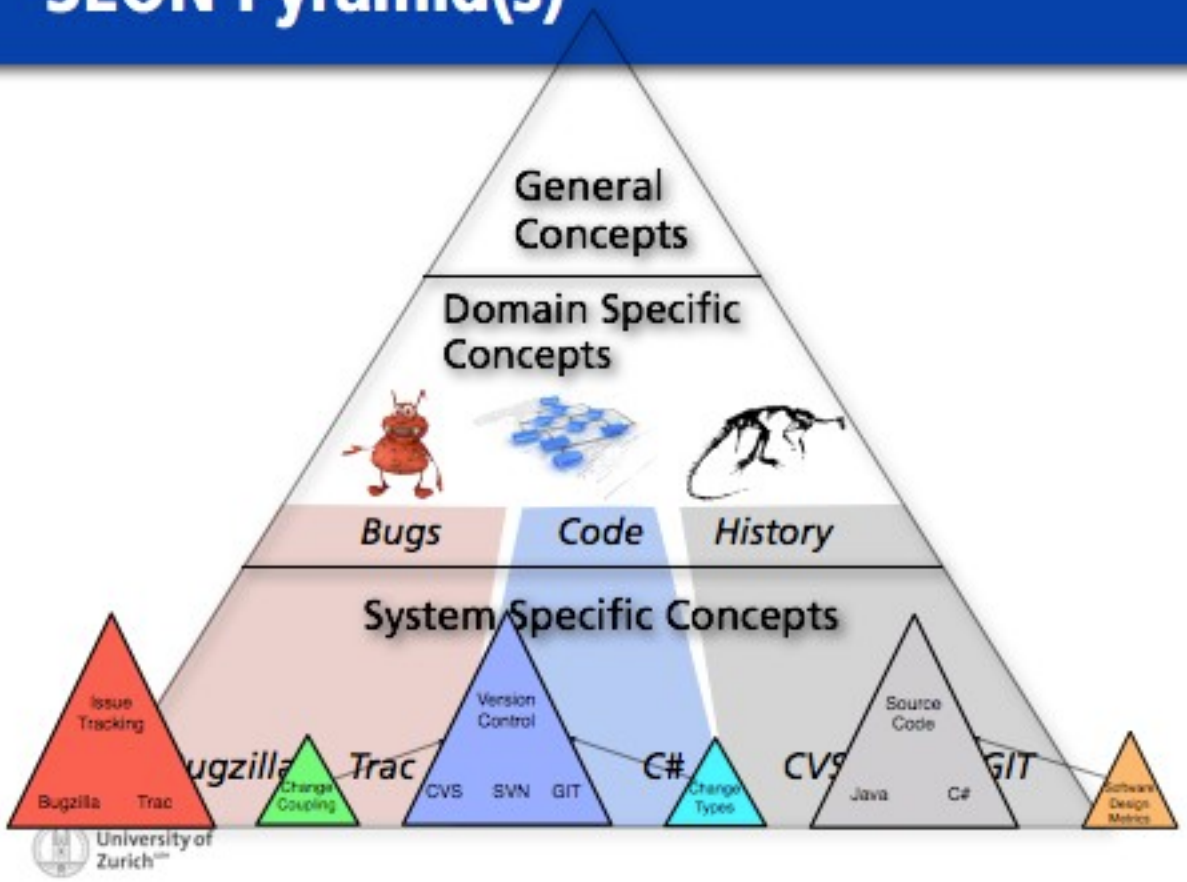


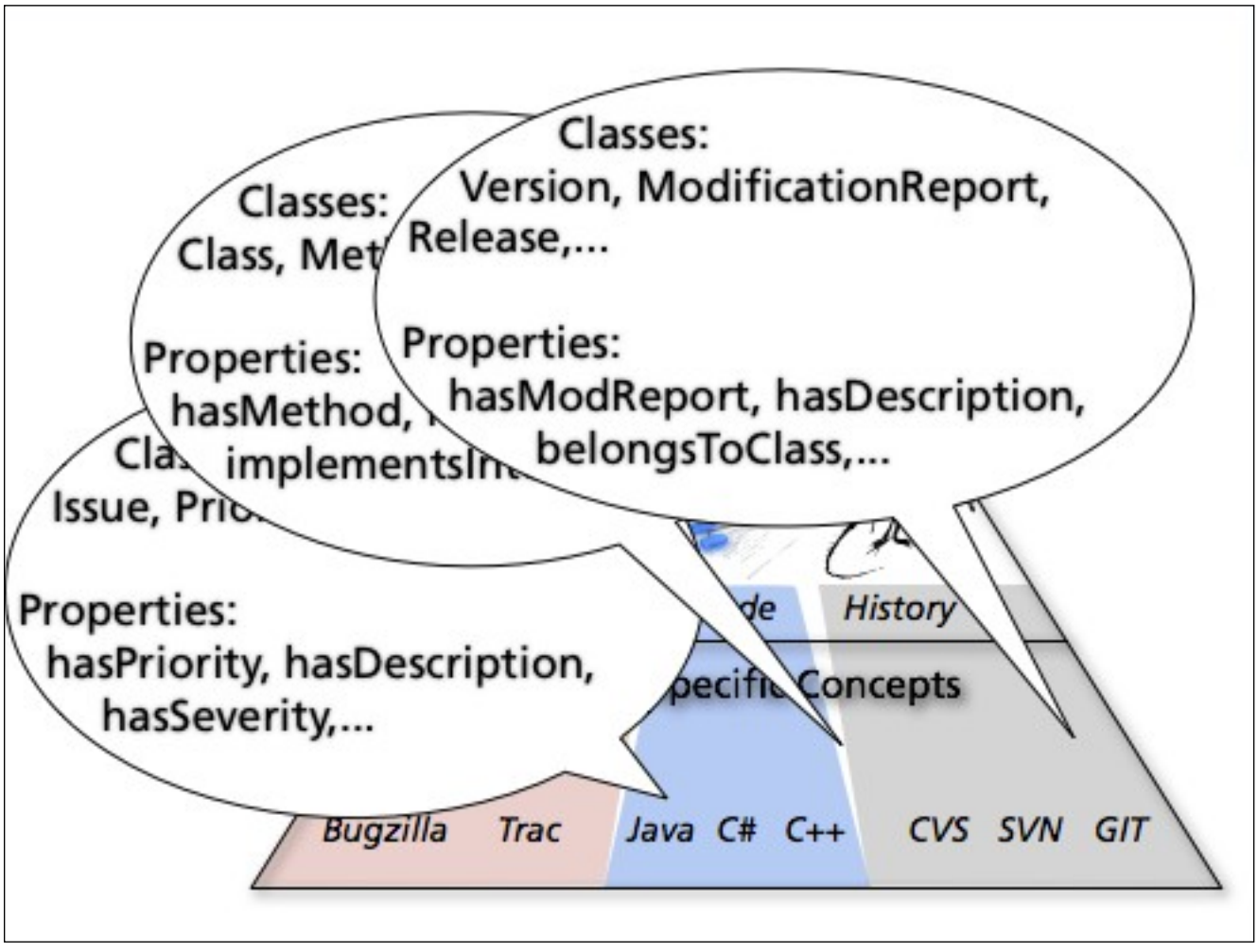
SOFAS scenario

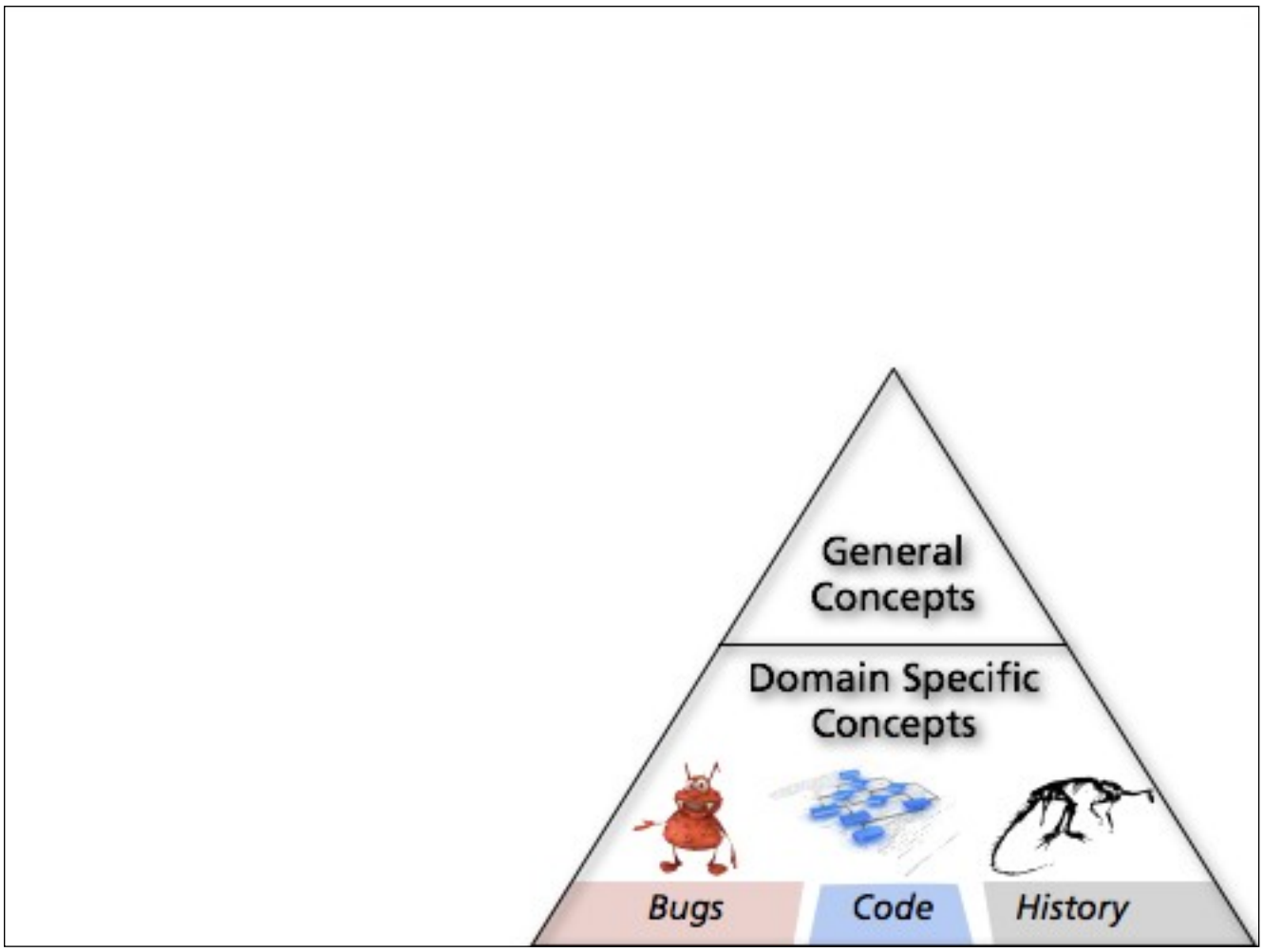


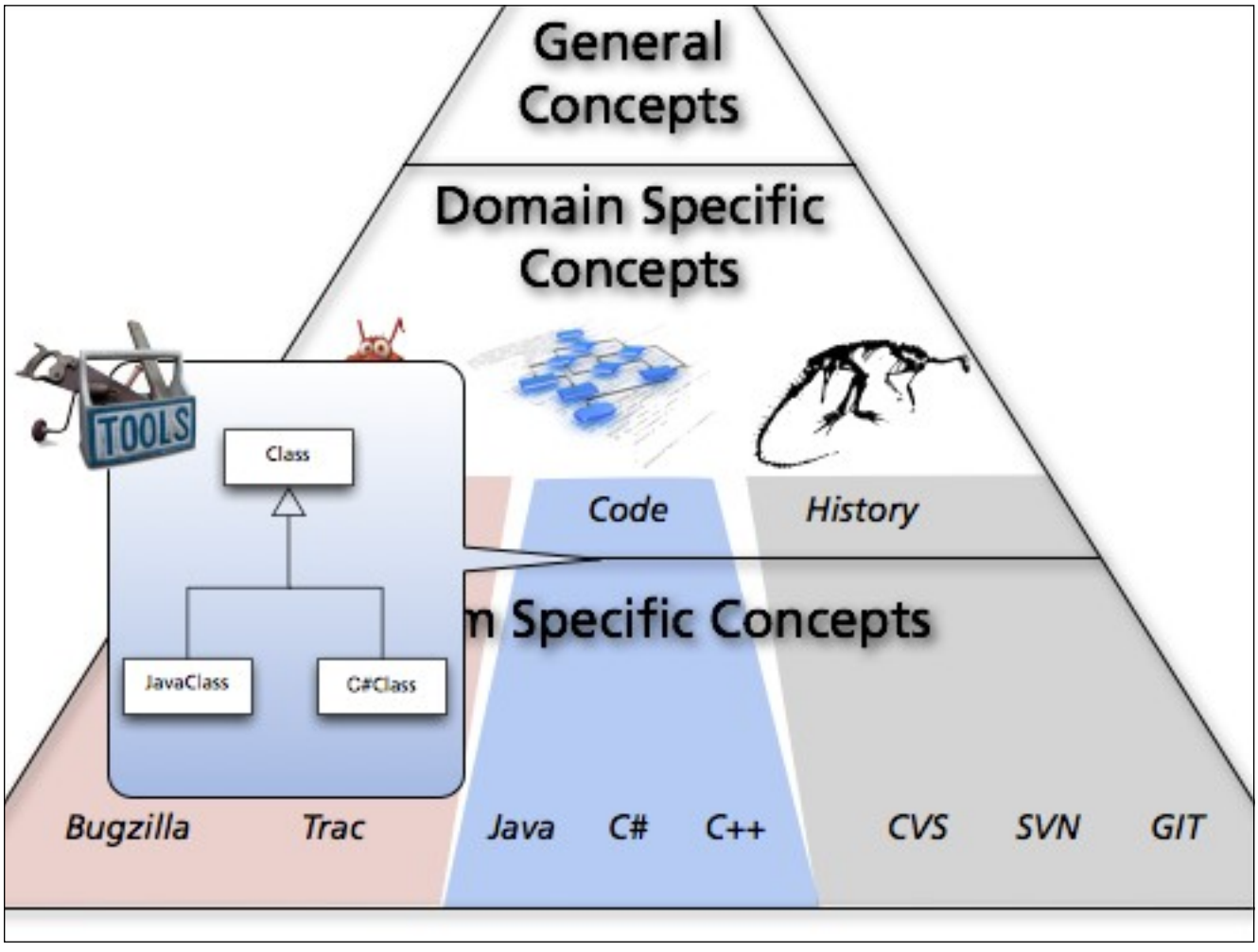


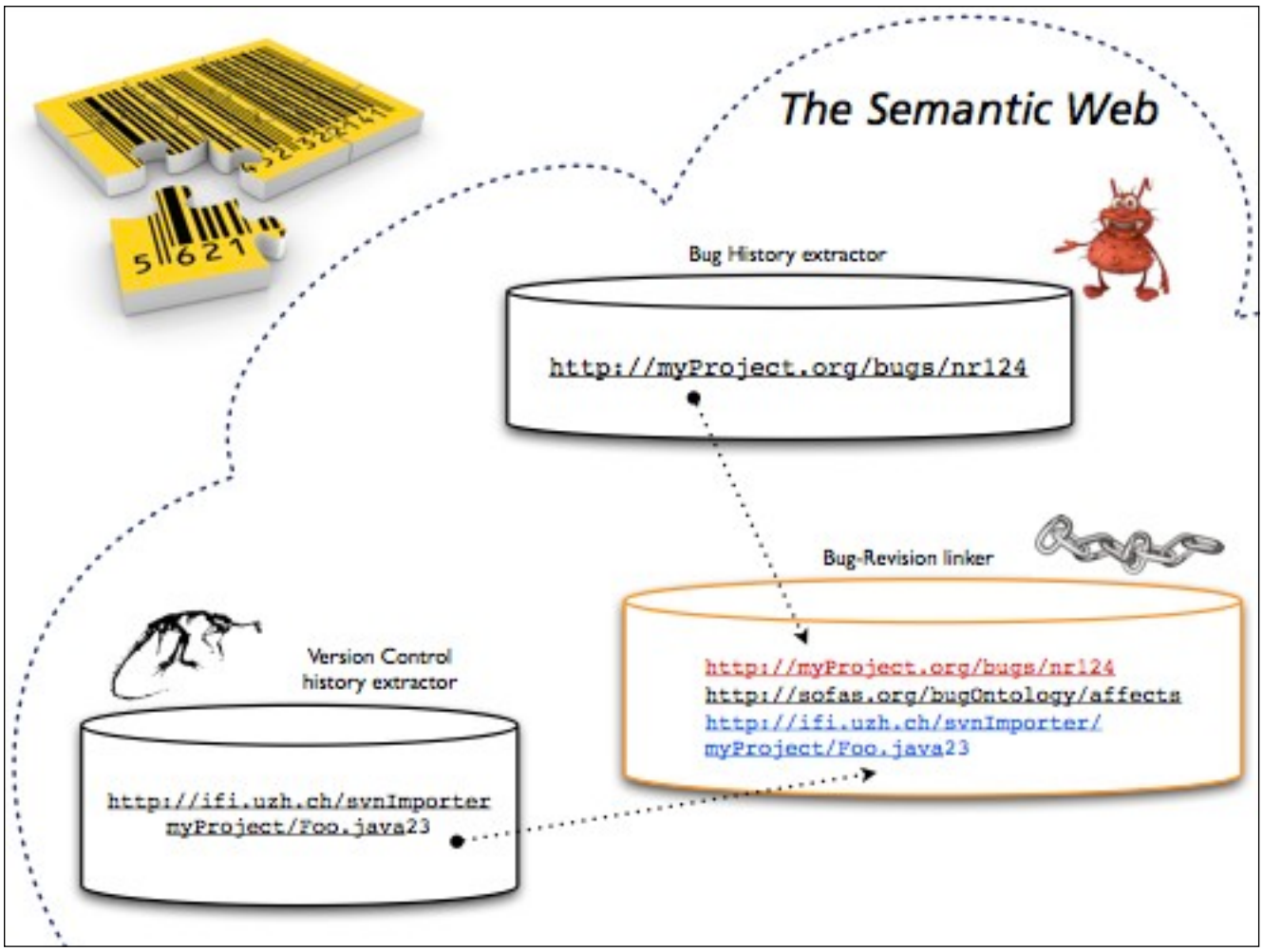
SEON Pyramid(s)






General
Concepts
Domain Specific
Concepts
Bugs
Code
History


General
Concepts
Domain Specific
Concepts
TOOLS
Class
JavaClass
C#Class
Code
History
Bugzilla
Trac
Java
C#
C++
CVS
SVN
GIT



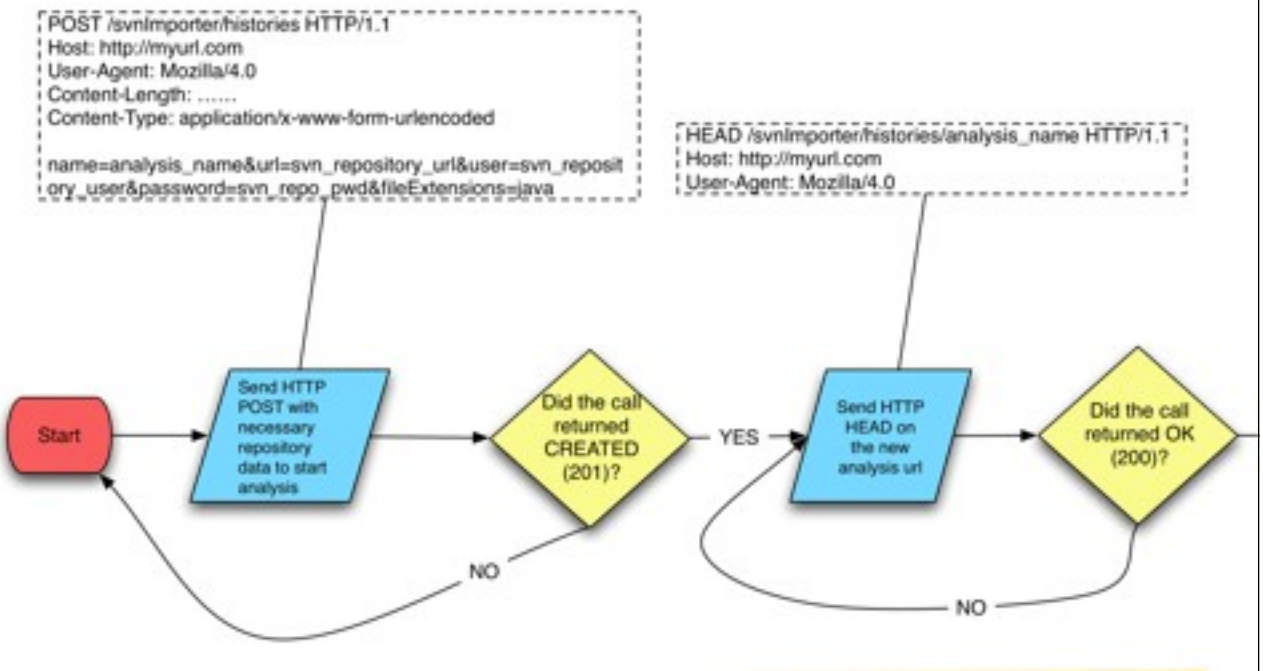
How to use SOFAS' RESTful services (1)

- Starting an analysis
  - access via HTTP, cURL, or Java/Jersey
  - send a post request to http://myurl.com/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 notify when the analysis is completed (however, a program can periodically send an HTTP HEAD request on the analysis URL if needed)

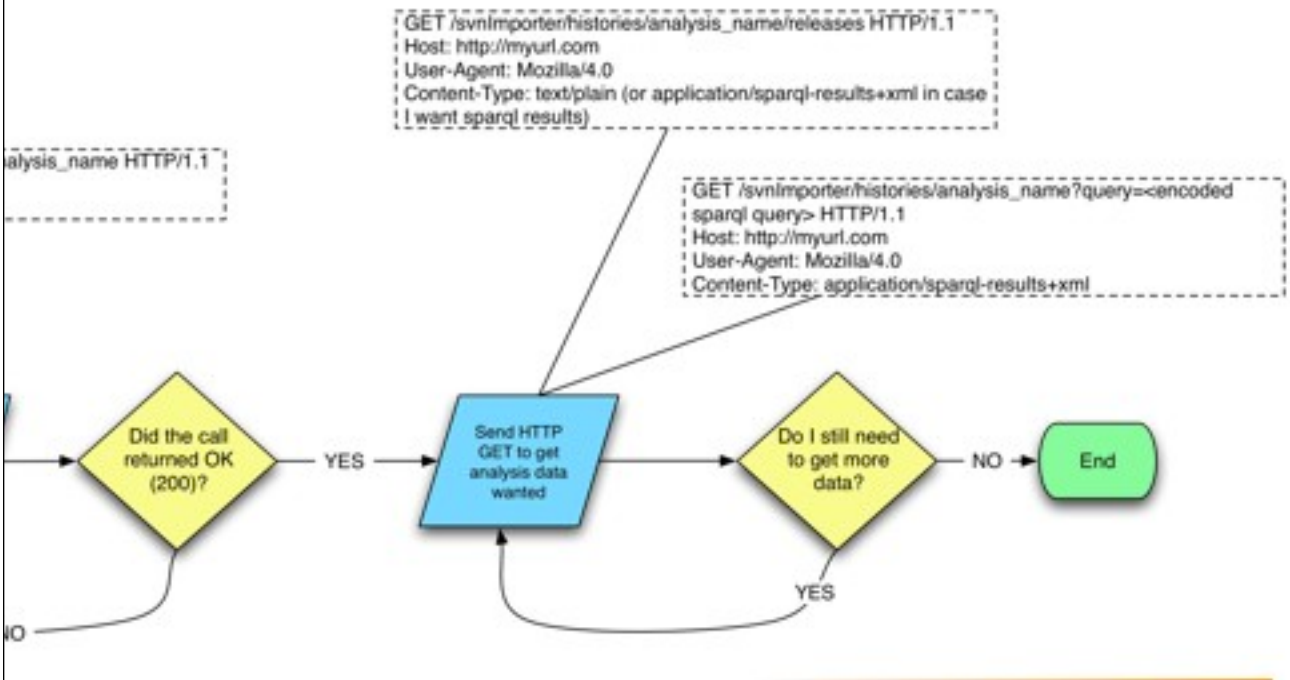
How to use SOFAS' RESTful services (2)

- 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)



SOFAS flow (2)



Current SOFAS services

- Version history extractor services for CVS, SVN, and GIT
- Meta-model extraction service (FAMIX)
- Issue tracking history services for Bugzilla, Google Code, Trac, and SourceForge
- Change Coupling service
- Change type distilling service
- Issue-revision linker services
- OO Metrics service

Composing Analysis Services

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

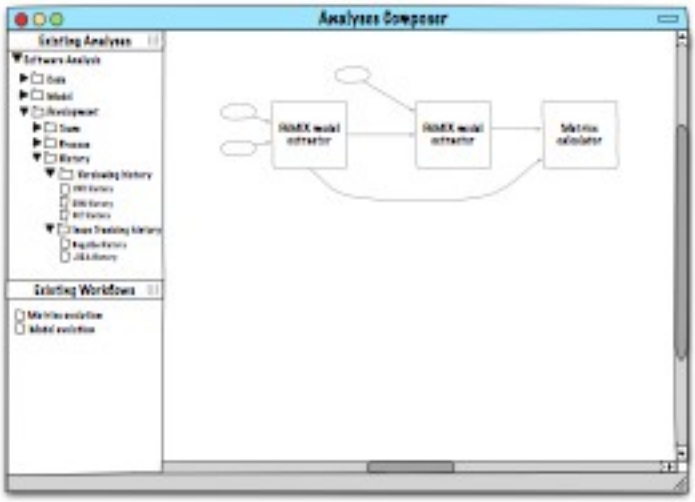
Software analysis composition

- Services should be (semi)-automatically composable depending on the analysis they offer
- Not just plain service composition but "semantically-supported" service composition

Software analysis composition

Composition, what to do?

- Define framework of software analysis composition
- Integrate with workflow technologies
- Create a web interface:
  - To easily compose different services into workflows
  - To use and modify preset workflow blue prints



Architecture Analysis on a Surface Table

s.e.a.l.

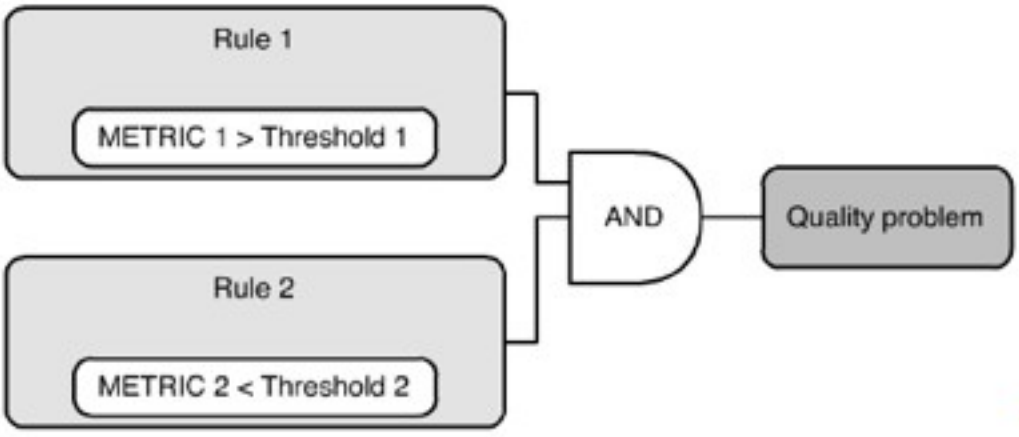
Multi-touch interfaces

- Multi-touch interfaces are widely used in various domains
- Little research on how it can be used in the context of SE

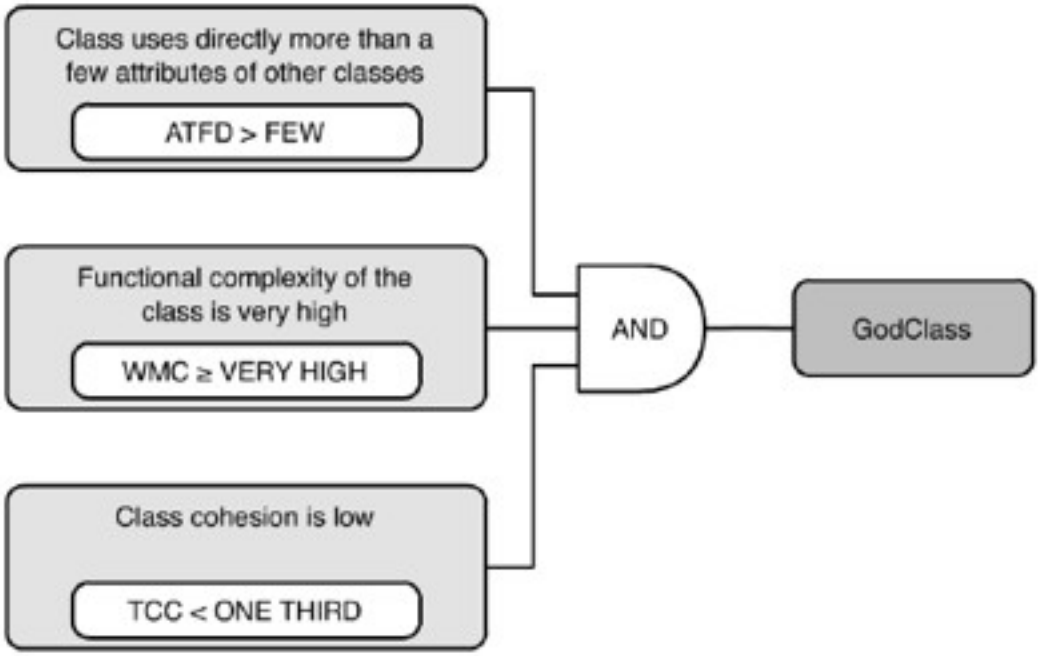
Microsoft Surface

Metrics Code Smell

- „Composed logical condition, based on metrics, that identifies those design fragments that are failing the condition."



God Class



University of Zurich

Tool: SmellTagger

s.e.a.l.

Key Features

- Scale, rotate and translate almost every UI element
- Mark and annotate part of the source code
- Take snapshots and email them or move them to the desktop
- Duplicate interesting content
- Tag objects

Tagged Objects

Some Visions

Ideas for new IDEs

- New IDEs should
  - integrate SA
  - use the browser as the IDE
  - have the IDE as the interface to comprehensive analyses
  - exploit visual (natural) UIs
  - make use of Data Warehouses

Project Looking Glass by Sun

NUIs for Software Designers

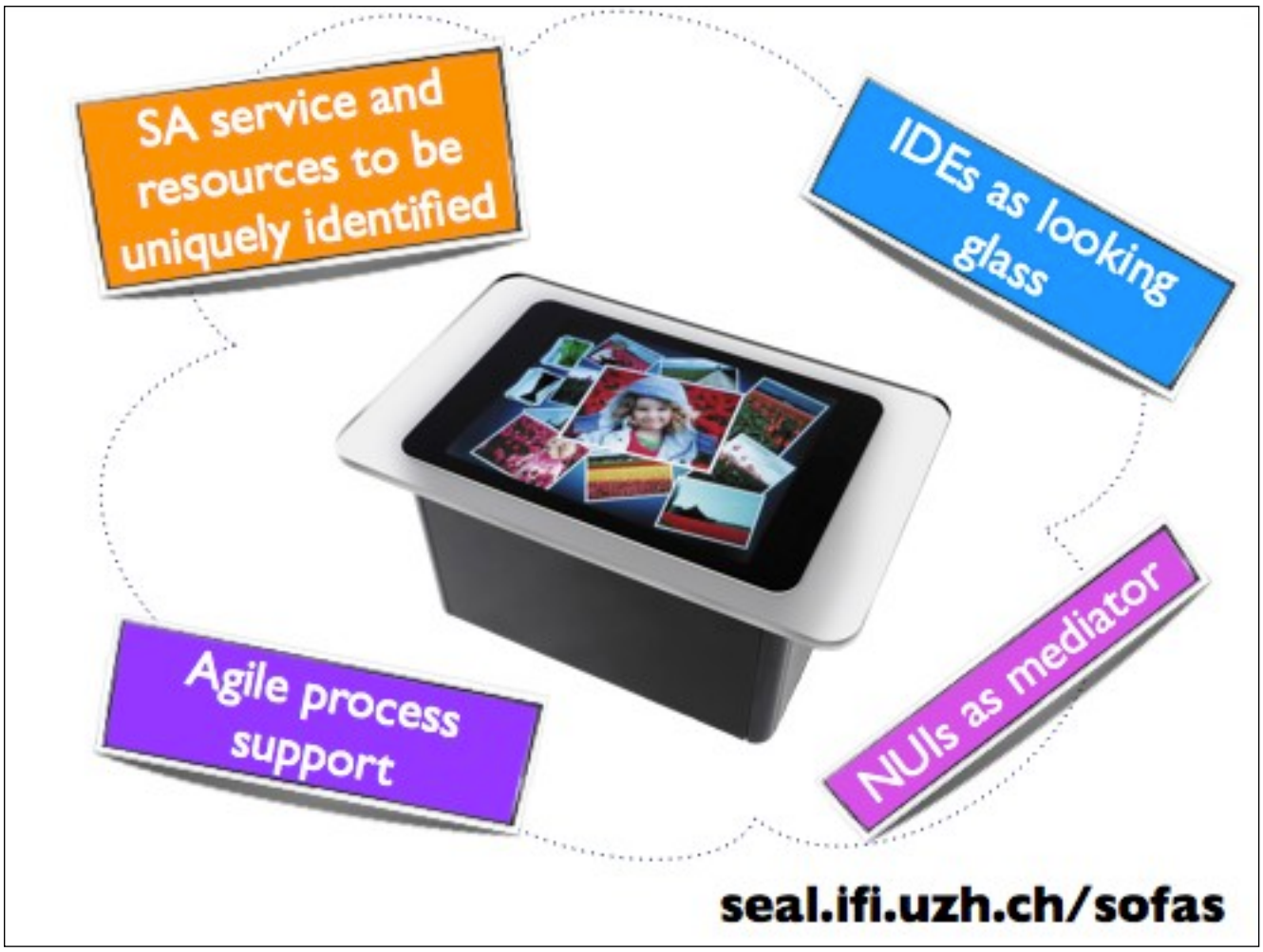
- Sensing software with
  - multitouch interfaces
  - haptic elements
  - cognitive metaphors
  - audio, video

Supporting teamwork

- New process support
  - support agile
  - be touch-enabled
  - by cognitive elements
- Scenarios
  - Stand-up meetings
  - Design reviews
  - Architecture analysis
  - Evolution Analysis
  - ...

The Cloud is a Market for SA

- Software analyses (SA) everywhere
- SA as multi-user backbone
- SA as a marketplace
  - service offerings
  - distributed, accessible
  - in the cloud


SA service and resources to be uniquely identified
IDEs as looking glass
Agile process support
NUIs as mediator
seal.ifi.uzh.ch/sofas