Identification and visualization of software architectures

Author: Bc. Marek Dinka
Supervisors: Ing. Martin Marko
doc. Ing. Ivan Polášek, PhD.

Aim

Design and create a prototype of the toolset capable of reverse engineering large and real software systems to classify and identify architecturally important component and their relations.

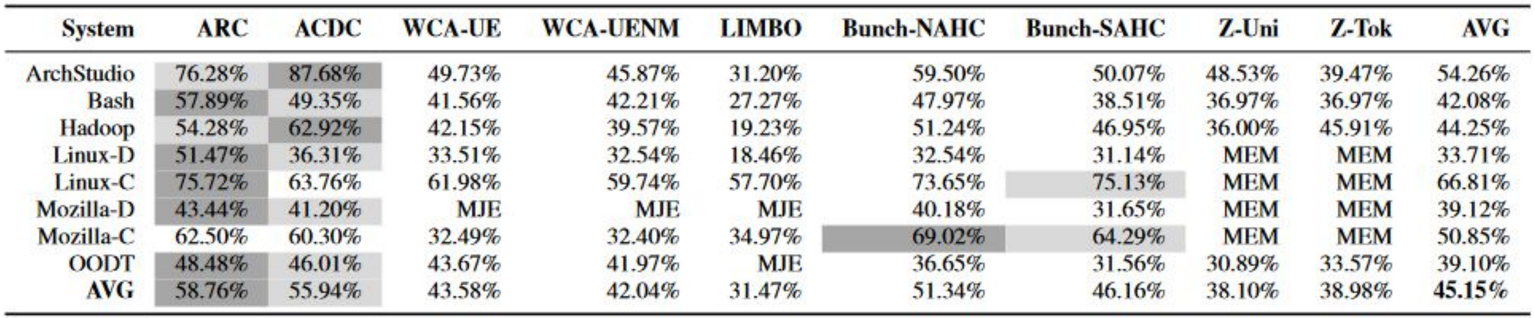
Propose **methods** and implement basic concept to use extracted information and derived relations. Document extracted information in the form of textual and visual architectural views.

Analysis of Architecture Recovery Techniques [5]

Comparative analysis of six automated architecture recovery techniques - ACDC, ARC, Bunch, LIMBO, WCA, ZBR

MoJoFM - distance measure between two architectures expressed as a percentage.

TABLE II: MoJoFM results



System	ARC	ACDC	WCA-UE	WCA-UENM	LIMBO	Bunch-NAHC	Bunch-SAHC	Z-Uni	Z-Tok	AVG
ArchStudio	76.28%	87.68%	49.73%	45.87%	31.20%	59.50%	50.07%	48.53%	39.47%	54.26%
Bash	57.89%	49.35%	41.56%	42.21%	27.27%	47.97%	38.51%	36.97%	36.97%	42.08%
Hadoop	54.28%	62.92%	42.15%	39.57%	19.23%	51.24%	46.95%	36.00%	45.91%	44.25%
Linux-D	51.47%	36.31%	33.51%	32.54%	18.46%	32.54%	31.14%	MEM	MEM	33.71%
Linux-C	75.72%	63.76%	61.98%	59.74%	57.70%	73.65%	75.13%	MEM	MEM	66.81%
Mozilla-D	43.44%	41.20%	MJE	MJE	MJE	40.18%	31.65%	MEM	MEM	39.12%
Mozilla-C	62.50%	60.30%	32.49%	32.40%	34.97%	69.02%	64.29%	MEM	MEM	50.85%
OODT	48.48%	46.01%	43.67%	41.97%	MJE	36.65%	31.56%	30.89%	33.57%	39.10%
AVG	58.76%	55.94%	43.58%	42.04%	31.47%	51.34%	46.16%	38.10%	38.98%	**45.15%**

[5] Joshua Garcia, Igor Ivkovic, and Nenad Medvidovic. A comparative analysis of software architecture recovery techniques. 2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE), pages 486–496, 2013.

Architecture Recovery using Cluster Ensembles [4]

Cluster ensemble is an approach of combining different clustering results into a single consolidated result

Analysis of multiple clustering techniques, comparison of their results, methods for consolidation

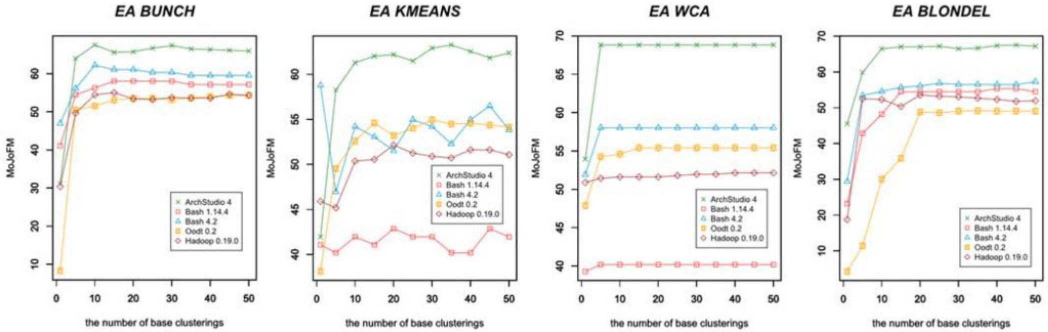


FIGURE 7. MoJoFM results of cluster ensemble-based recovery methods on varying numbers of base clusterings. The used sets of base clusterings are sets 1 to 4 in Fig. 5 and the applied consensus method is the EA

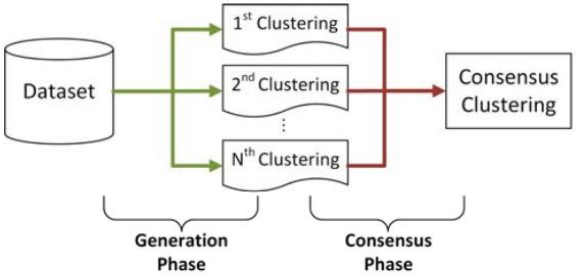


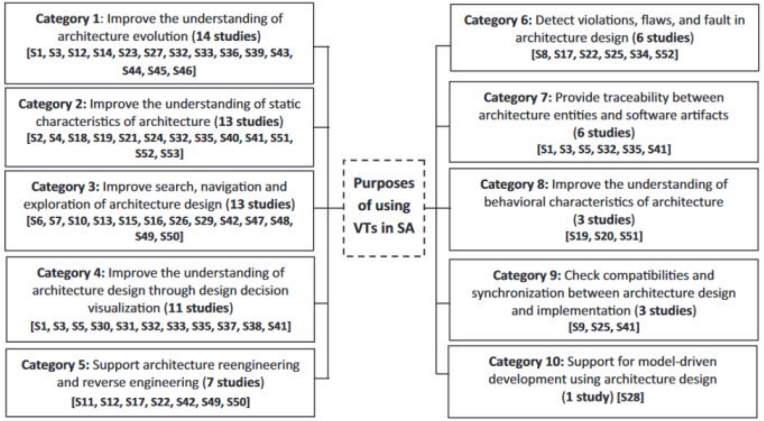
FIGURE 1. The general process of the cluster ensemble technique consists of two phases: generation and consensus.

[4] Choongki Cho, Ki-Seong Lee, Minsoo Lee, and Chan-Gun Lee. Software architecture module-view recovery using cluster ensembles. IEEE Access, 7:72872–72884, 2019.

Review of Architecture Visualization Techniques [33]

Overview of recent works in software architecture visualization

Categorization of architecture visualization techniques and purposes



Category 1: Improve the understanding of architecture evolution (**14 studies**) [S1, S3, S12, S14, S23, S27, S32, S33, S36, S39, S43, S44, S45, S46]

Category 2: Improve the understanding of static characteristics of architecture (**13 studies**) [S2, S4, S18, S19, S21, S24, S32, S35, S40, S41, S51, S52, S53]

Category 3: Improve search, navigation and exploration of architecture design (**13 studies**) [S6, S7, S10, S13, S15, S16, S26, S29, S42, S47, S48, S49, S50]

Category 4: Improve the understanding of architecture design through design decision visualization (**11 studies**) [S1, S3, S5, S30, S31, S32, S33, S35, S37, S38, S41]

Category 5: Support architecture reengineering and reverse engineering (**7 studies**) [S11, S12, S17, S22, S42, S49, S50]

Purposes of using VTs in SA

Category 6: Detect violations, flaws, and fault in architecture design (**6 studies**) [S8, S17, S22, S25, S34, S52]

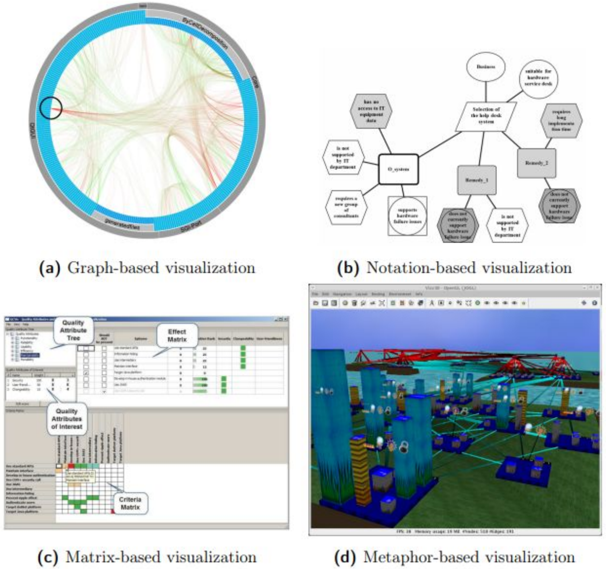
Category 7: Provide traceability between architecture entities and software artifacts (**6 studies**) [S1, S3, S5, S32, S35, S41]

Category 8: Improve the understanding of behavioral characteristics of architecture (**3 studies**) [S19, S20, S51]

Category 9: Check compatibilities and synchronization between architecture design and implementation (**3 studies**) [S9, S25, S41]

Category 10: Support for model-driven development using architecture design (**1 study**) [S28]

Categorization of architecture visualization purposes



(a) Graph-based visualization

(b) Notation-based visualization

(c) Matrix-based visualization

(d) Metaphor-based visualization

Categorization of architecture visualization techniques

[29] Mojtaba Shahin, Peng Liang, and Muhammad Ali Babar. A systematic review of software architecture visualization techniques. Journal of Systems and Software, 94:161–185, 2014.

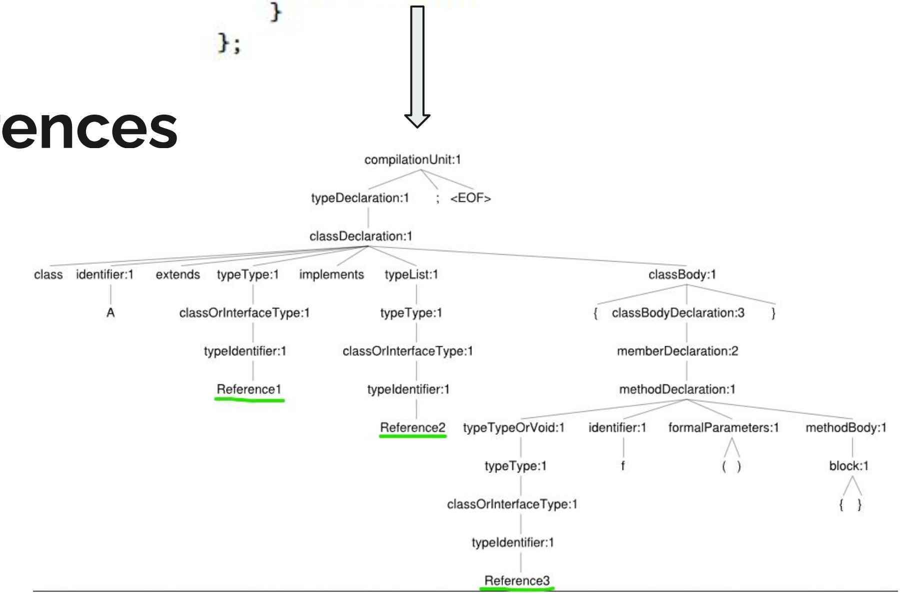
Obtaining Interclass References

A simple starting point

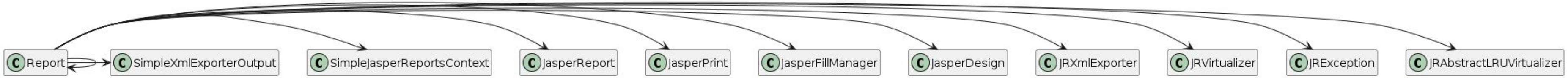
Use a parsing tool for the java language to identify all classes referenced in a file.

Helps with recognizing relations, dependencies, path of execution, …

```
class A extends Reference1 implements Reference2 {
    Reference3 f() {
    }
};
```



Parsed deconstruction of a java class named A



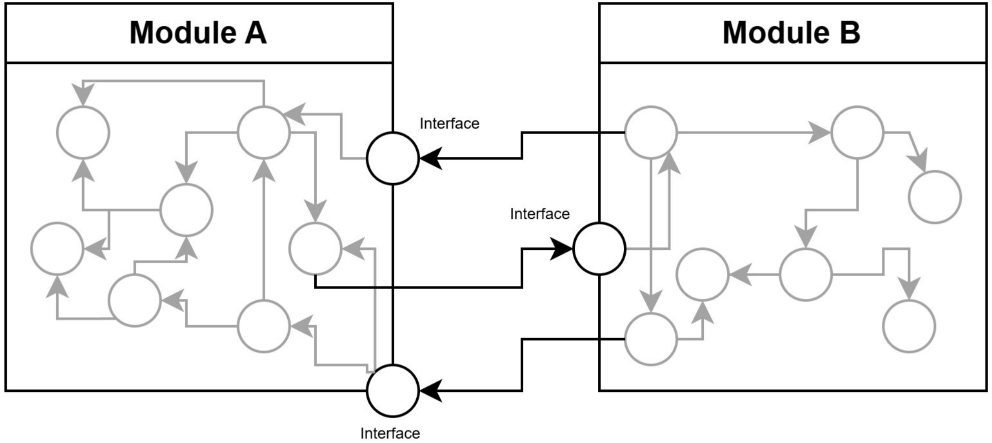
Classes referenced in the Report class of Jasperreports project

Modular Decomposition

A module is a self-contained unit of code with well defined interfaces and specific task(s) [31]

Idea: Class should not reference a class from another module, unless this class is an API of said module and API classes should be few

Clustering based on common class references

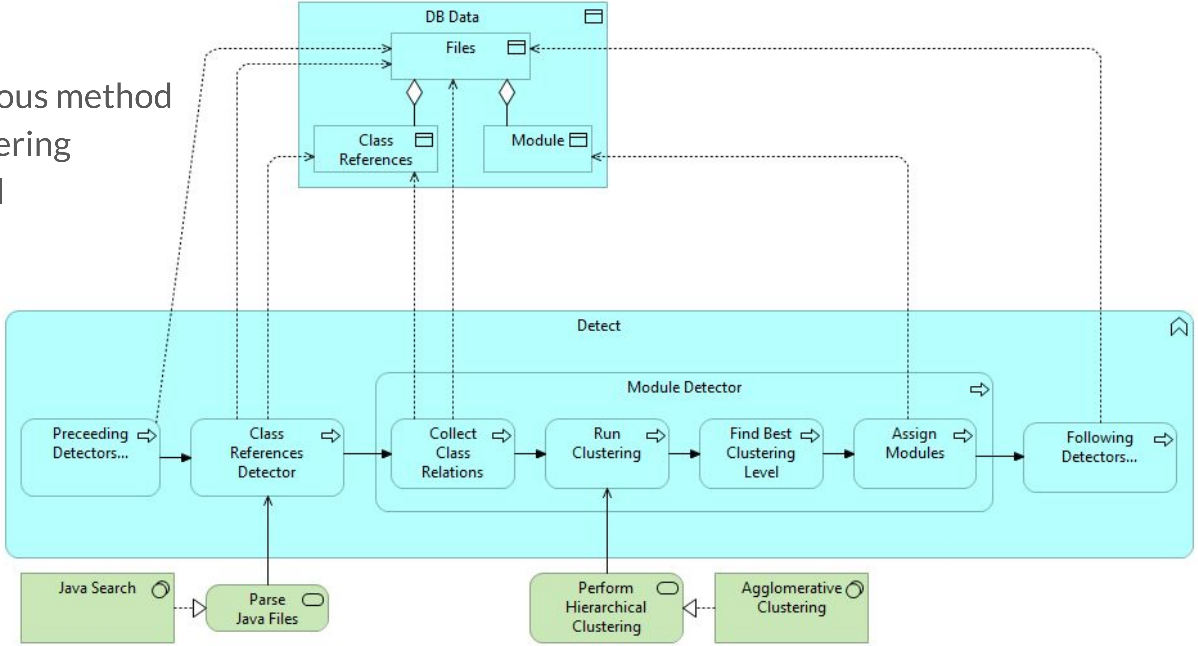


A diagram illustrating the ideal modular architecture

[31] D. L. Parnas. On the criteria to be used in decomposing systems into modules. Commun. ACM, 15(12):1053–1058, December 1972.

Modular Decomposition

1. Use references from previous method
2. Perform hierarchical clustering
3. Find best hierarchical level
4. Assign modules to files



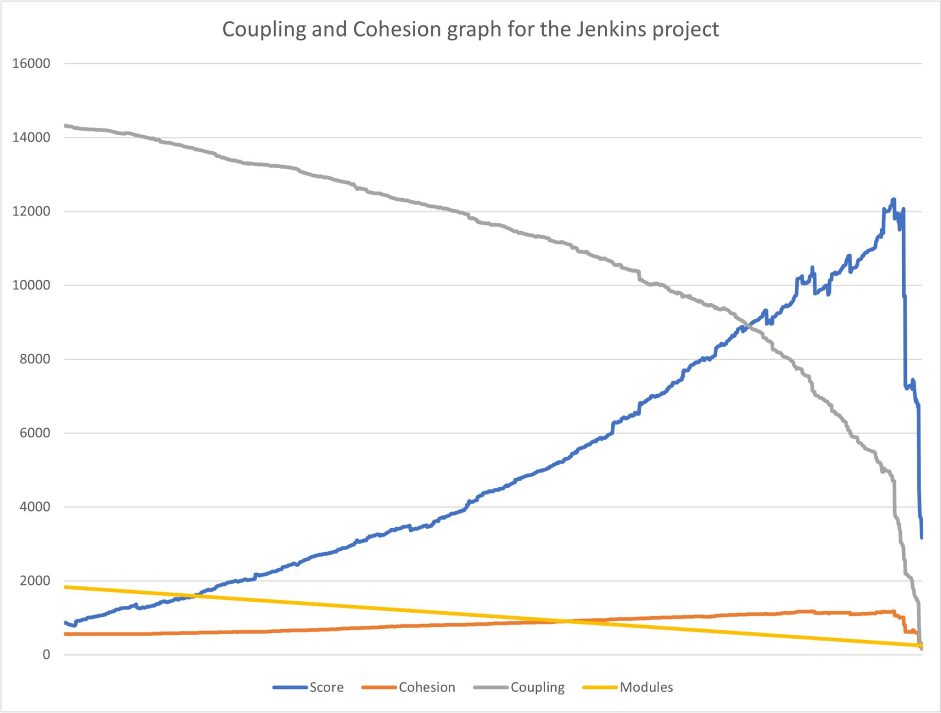
Archimate model describing the architecture behind clustering of modules

Modular Decomposition

Coupling describes connectivity among subsystems.
Cohesion describes connectivity within subsystems.

Designs with low coupling and high cohesion lead to products that are both, more reliable and more maintainable. [20]

Best clustering level = best combination of coupling and cohesion + reasonable amount of modules
(i.e. not 1)



Coupling and Cohesion graph for the Jenkins project

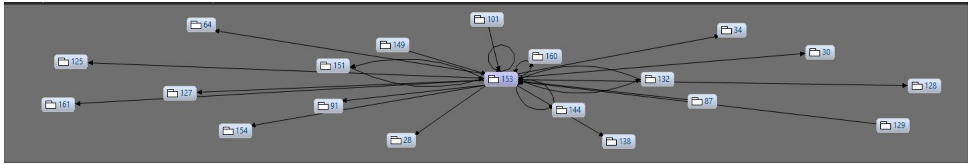
[20] Martin Hitz and Behzad Montazeri. Measuring coupling and cohesion in object oriented systems..

Modular Decomposition

Visualization of standalone modules - difficult

Further investigation of visualization methods required

Further investigation of ways for identifying module’s function required



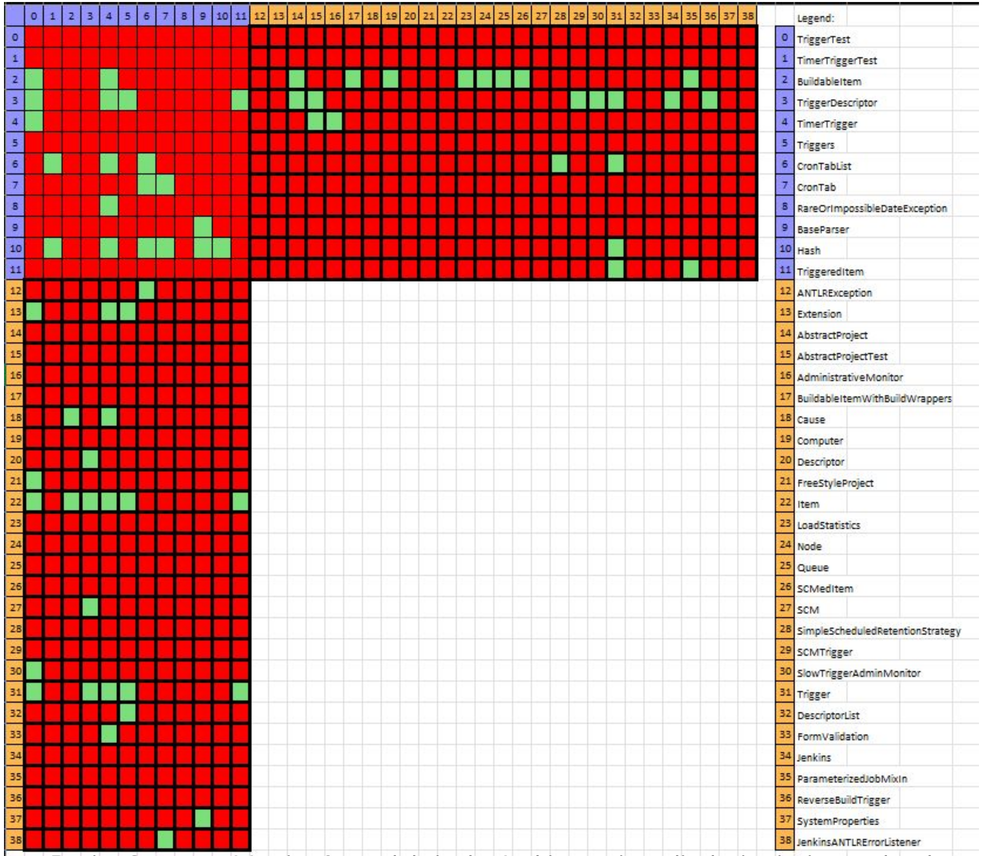
Relations of a Module drawn using archimate visualizer

All classes of the Jenkins project divided into modules

Design Structure Matrix

Design structure matrix is a network modeling tool used to represent the elements compromising a system and their interactions, thereby highlighting the system's architecture [16].

A visualization method used for judging the quality of modules (in context of cohesion and coupling)



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
0	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
1	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
2	🟩	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟩	🟥	🟩	🟥	🟥	🟥	🟩	🟩	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟥
3	🟩	🟥	🟥	🟥	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟩	🟩	🟥	🟥	🟩	🟥	🟩	🟥	🟥
4	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
5	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
6	🟥	🟩	🟥	🟥	🟩	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥
7	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
8	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
9	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
10	🟥	🟩	🟥	🟥	🟩	🟥	🟩	🟩	🟥	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥
11	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟩	🟥	🟥	🟥
12	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥																											
13	🟩	🟥	🟥	🟥	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟥																											
14	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
15	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
16	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
17	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
18	🟥	🟥	🟩	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
19	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
20	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
21	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
22	🟩	🟥	🟩	🟩	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟩																											
23	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
24	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
25	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
26	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
27	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
28	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
29	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
30	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
31	🟩	🟥	🟥	🟩	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟩																											
32	🟥	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥																											
33	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
34	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
35	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
36	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥																											
37	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟥	🟥																											
38	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟥	🟥	🟥	🟥																											

Legend:

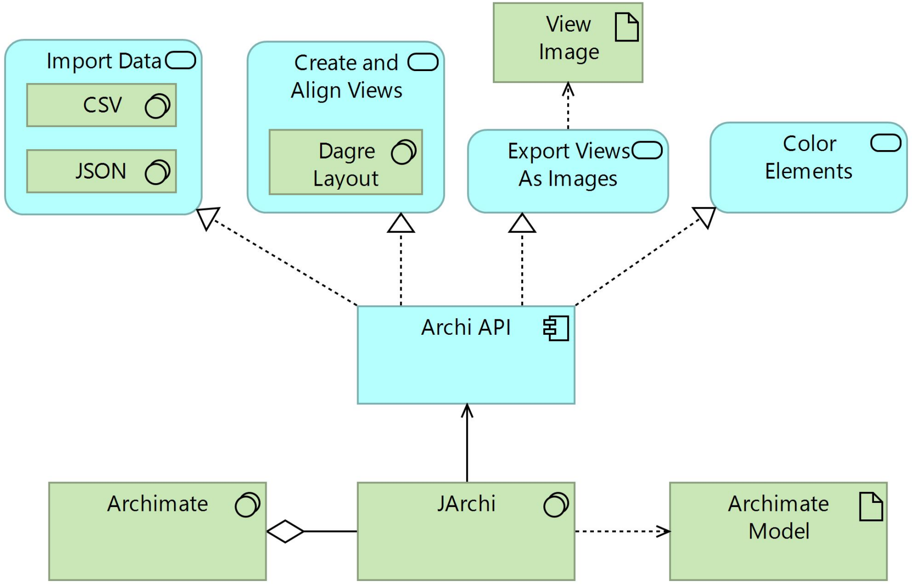
0	TriggerTest
1	TimerTriggerTest
2	BuildableItem
3	TriggerDescriptor
4	TimerTrigger
5	Triggers
6	CronTabList
7	CronTab
8	RareOrImpossibleDateException
9	BaseParser
10	Hash
11	TriggeredItem
12	ANTLRException
13	Extension
14	AbstractProject
15	AbstractProjectTest
16	AdministrativeMonitor
17	BuildableItemWithBuildWrappers
18	Cause
19	Computer
20	Descriptor
21	FreeStyleProject
22	Item
23	LoadStatistics
24	Node
25	Queue
26	SCMedItem
27	SCM
28	SimpleScheduledRetentionStrategy
29	SCMTrigger
30	SlowTriggerAdminMonitor
31	Trigger
32	DescriptorList
33	FormValidation
34	Jenkins
35	ParameterizedJobMixIn
36	ReverseBuildTrigger
37	SystemProperties
38	JenkinsANTLRErrorListener

Design Structure Matrix of a module in the Jenkins project, displaying its internal and external class relations

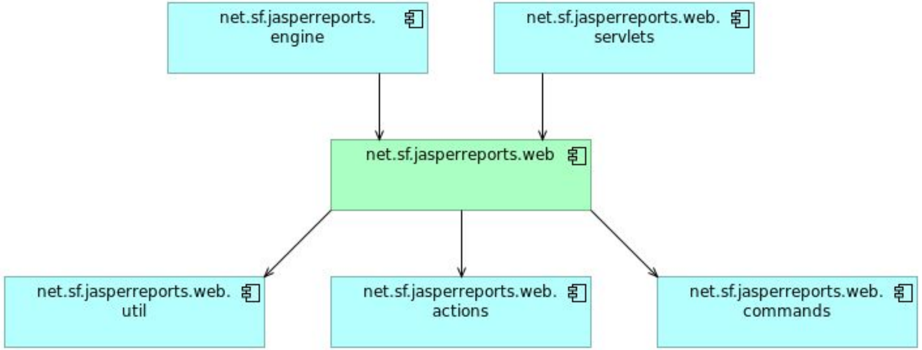
[16] SD Eppinger. Design Structure Matrix Methods and Applications. MIT Press, 2012.

Archimate CLI API

We use Archimate's scripting plugin (JArchi) to create a CLI interface for interacting with it's models



Archimate diagram displaying the architecture of Archi API

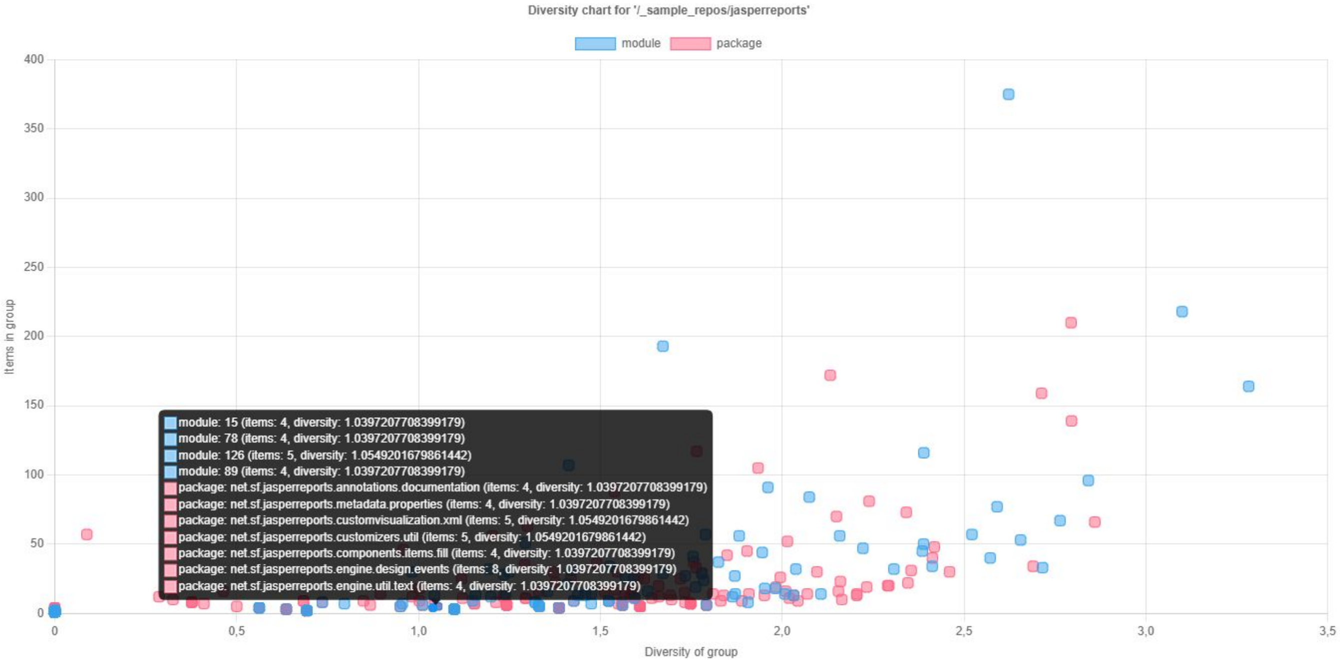

net.sf.jasperreports.engine
net.sf.jasperreports.web.servlets
net.sf.jasperreports.web
net.sf.jasperreports.web.util
net.sf.jasperreports.web.actions
net.sf.jasperreports.web.commands

Dependencies of the web package of the Jasperreports project, generated using Archi Api

File Diversity Chart

Comparison of different groupings (e.g. modules, java packages, build structure) based on the diversity of files within said grouping

Interactive chart generated using Shannon's or Simpson's diversity index

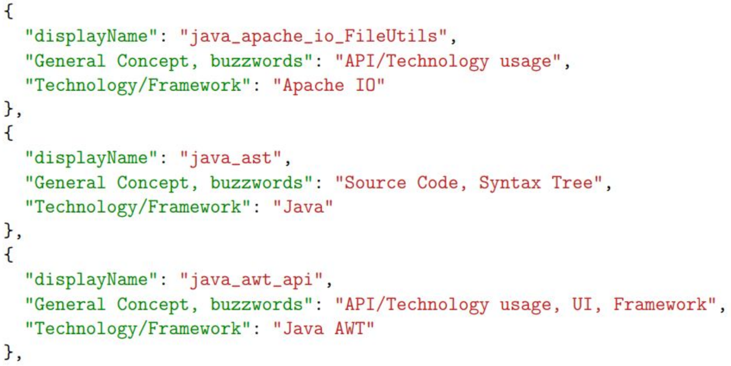


Diversity chart comparing modules and packages for the Jasperreports project

File Descriptors as Genomes

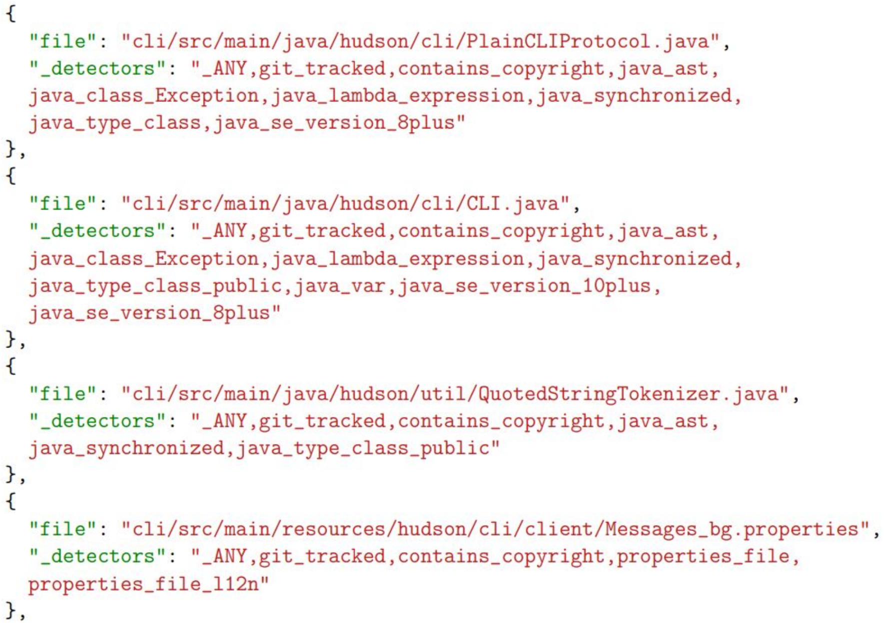
Each file analyzed by Cinderella can be described by a list of detectors, each detector can additionally be described by a list of attributes

We will run clustering based on these attributes



```
{
  "displayName": "java_apache_io_FileUtils",
  "General Concept, buzzwords": "API/Technology usage",
  "Technology/Framework": "Apache IO"
},
{
  "displayName": "java_ast",
  "General Concept, buzzwords": "Source Code, Syntax Tree",
  "Technology/Framework": "Java"
},
{
  "displayName": "java_awt_api",
  "General Concept, buzzwords": "API/Technology usage, UI, Framework",
  "Technology/Framework": "Java AWT"
},
```

Extract from the documentation of detectors



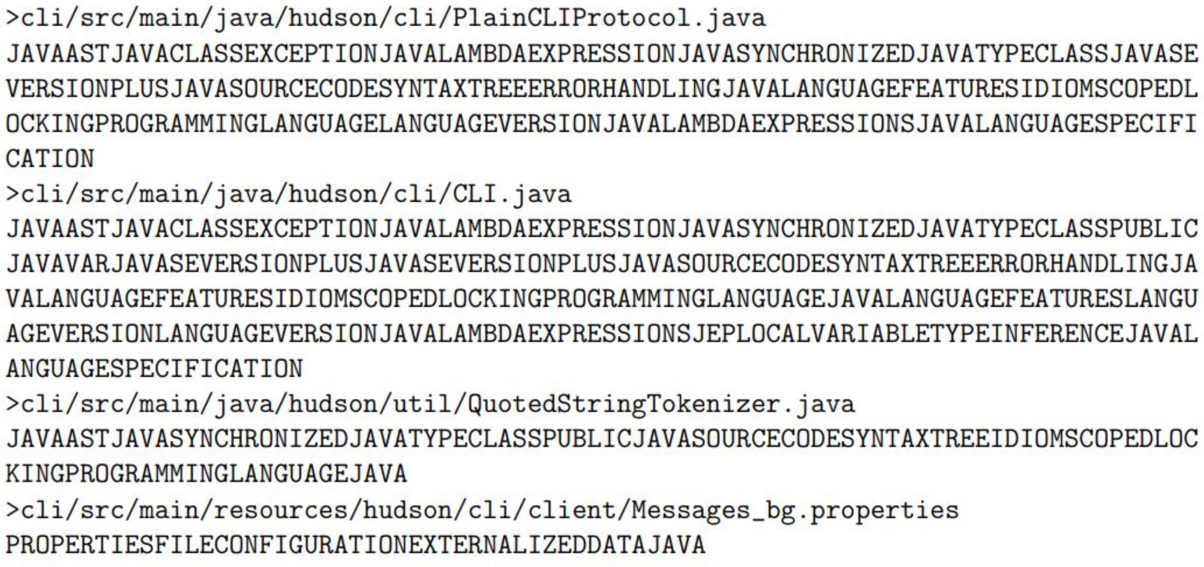
```
{
  "file": "cli/src/main/java/hudson/cli/PlainCLIProtocol.java",
  "_detectors": "_ANY,git_tracked,contains_copyright,java_ast,
  java_class_Exception,java_lambda_expression,java_synchronized,
  java_type_class,java_se_version_8plus"
},
{
  "file": "cli/src/main/java/hudson/cli/CLI.java",
  "_detectors": "_ANY,git_tracked,contains_copyright,java_ast,
  java_class_Exception,java_lambda_expression,java_synchronized,
  java_type_class_public,java_var,java_se_version_10plus,
  java_se_version_8plus"
},
{
  "file": "cli/src/main/java/hudson/util/QuotedStringTokenizer.java",
  "_detectors": "_ANY,git_tracked,contains_copyright,java_ast,
  java_synchronized,java_type_class_public"
},
{
  "file": "cli/src/main/resources/hudson/cli/client/Messages_bg.properties",
  "_detectors": "_ANY,git_tracked,contains_copyright,properties_file,
  properties_file_l12n"
},
```

Files of the Jenkins project together with detectors triggered on each of them

File Descriptors as Genomes

Multiple sequence alignment is a way of arranging the sequences of DNA, RNA, or protein to identify regions of similarity [36].

Clustal omega is a tool which can be used for performing fast MSAs of potentially large sequences



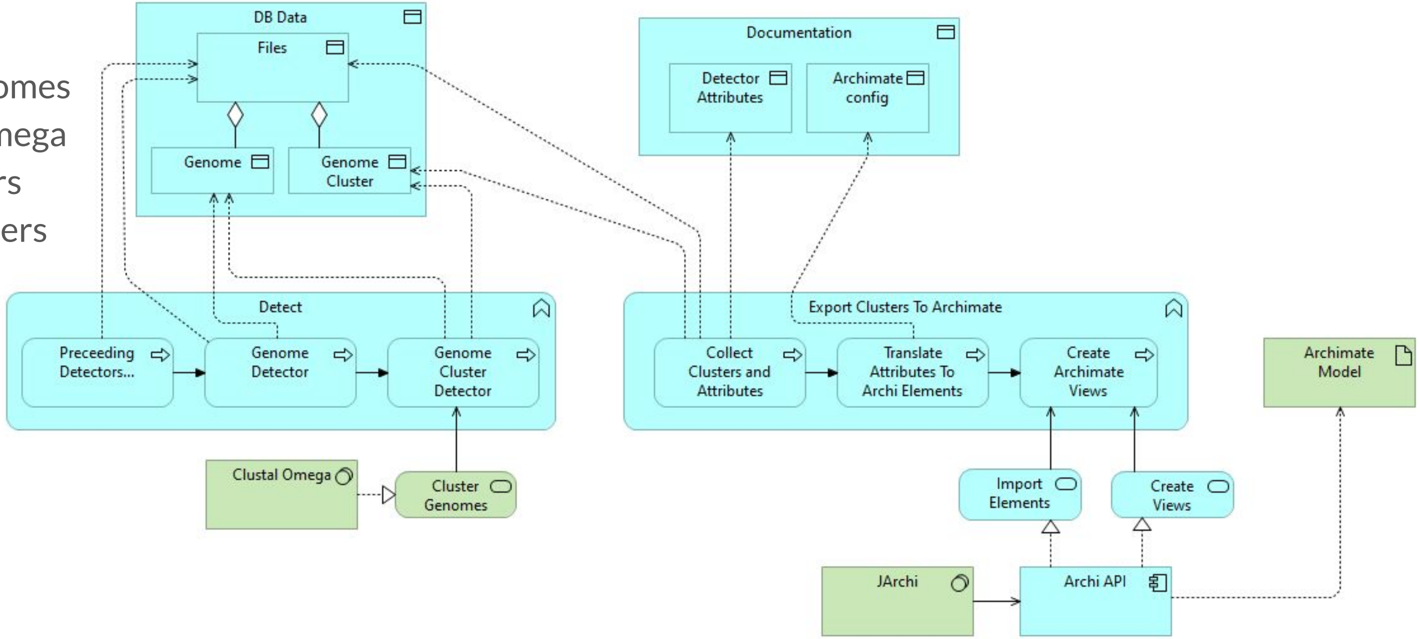
```
>cli/src/main/java/hudson/cli/PlainCLIProtocol.java
JAVAASTJAVACLASSEXCEPTIONJAVALAMBDAEXPRESSIONJAVASYNCHRONIZEDJAVATYPECLASSJAVASE
VERSIONPLUSJAVASOURCECODESYNTAXTREEERRORHANDLINGJAVALANGUAGEFEATURESIDIOMSCOPEDL
OCKINGPROGRAMMINGLANGUAGELANGUAGEVERSIONJAVALAMBDAEXPRESSIONSJAVALANGUAGESPECIFI
CATION
>cli/src/main/java/hudson/cli/CLI.java
JAVAASTJAVACLASSEXCEPTIONJAVALAMBDAEXPRESSIONJAVASYNCHRONIZEDJAVATYPECLASSPUBLIC
JAVAVARJAVASEVERSIONPLUSJAVASEVERSIONPLUSJAVASOURCECODESYNTAXTREEERRORHANDLINGJA
VALANGUAGEFEATURESIDIOMSCOPEDLOCKINGPROGRAMMINGLANGUAGEJAVALANGUAGEFEATURESLANGU
AGEVERSIONLANGUAGEVERSIONJAVALAMBDAEXPRESSIONSJEPLOCALVARIABLETYPEINFERENCEJAVAL
ANGUAGESPECIFICATION
>cli/src/main/java/hudson/util/QuotedStringTokenizer.java
JAVAASTJAVASYNCHRONIZEDJAVATYPECLASSPUBLICJAVASOURCECODESYNTAXTREEIDIOMSCOPEDLOC
KINGPROGRAMMINGLANGUAGEJAVA
>cli/src/main/resources/hudson/cli/client/Messages_bg.properties
PROPERTIESFILECONFIGURATIONEXTERNALIZEDDATAJAVA
```

Genomes generated for files in the Jenkins project (in FASTA format)

[36] Fabian Sievers and Desmond G Higgins. Clustal Omega, accurate alignment of very large numbers of sequences. Springer, 2014.

File Descriptors as Genomes

1. Generate genomes
2. Run Clustal omega
3. Extract clusters
4. Visualize clusters

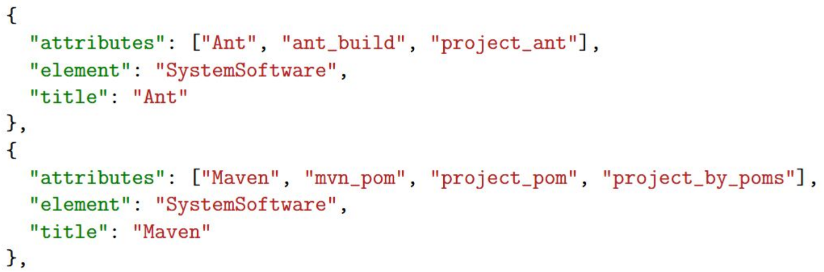


Archimate diagram displaying the architecture of genome clustering

File Descriptors as Genomes

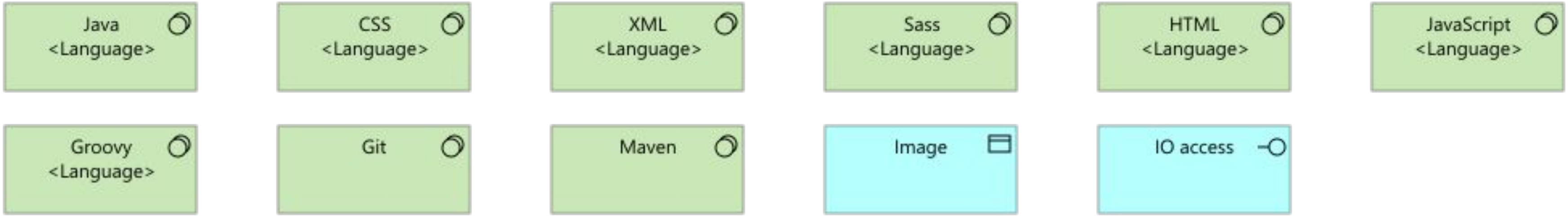
Visualization:

1. Take all commonly occurring attributes in a cluster
2. Translate these attributes into archimate elements
3. Draw the elements, thus linking files to ‘components’



```
{
  "attributes": ["Ant", "ant_build", "project_ant"],
  "element": "SystemSoftware",
  "title": "Ant"
},
{
  "attributes": ["Maven", "mvn_pom", "project_pom", "project_by_poms"],
  "element": "SystemSoftware",
  "title": "Maven"
},
```

Extract from JSON array responsible for mapping attributes to elements

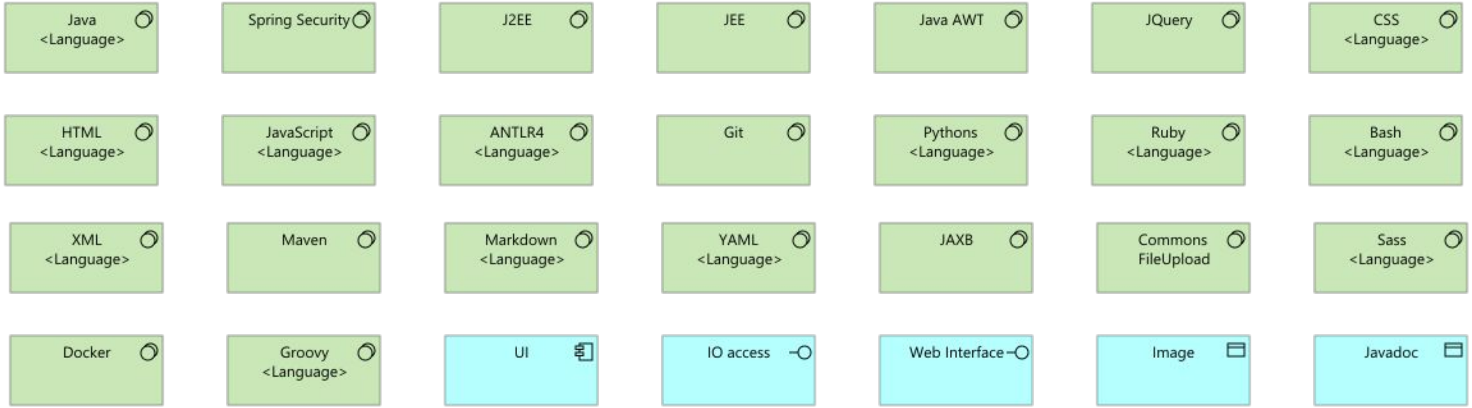


Elements generated from **clustered** attributes (note: not all attributes have yet been mapped)

Abstraction Context

Problem:

- Some attributes are more common than others
- The less common attributes might be drowned out by the more common ones
- The more important attributes are usually less common



Java <Language>

Spring Security

J2EE

JEE

Java AWT

JQuery

CSS <Language>

HTML <Language>

JavaScript <Language>

ANTLR4 <Language>

Git

Pythons <Language>

Ruby <Language>

Bash <Language>

XML <Language>

Maven

Markdown <Language>

YAML <Language>

JAXB

Commons FileUpload

Sass <Language>

Docker

Groovy <Language>

UI

IO access

Web Interface

Image

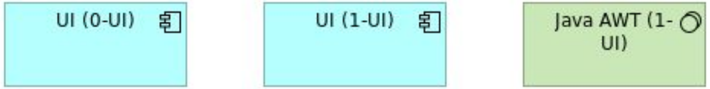
Javadoc

Elements generated from **all** attributes (note: not all attributes have yet been mapped)

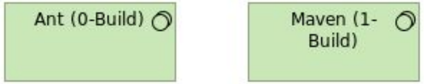
Abstraction Context

Solution:

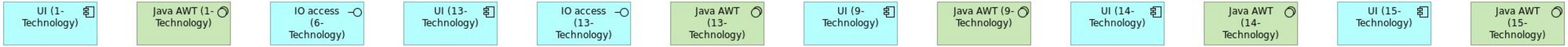
- Introduce concept of abstraction context (or detector context)
- Attributes will be assigned into groups based on the abstractions which they describe
- e.g. Build context, Technology context, UI context, …



UI context view extracted from the Jenkins project



Build context view extracted from the Jasperreports project



Technology context view extracted from the Jasperreports project