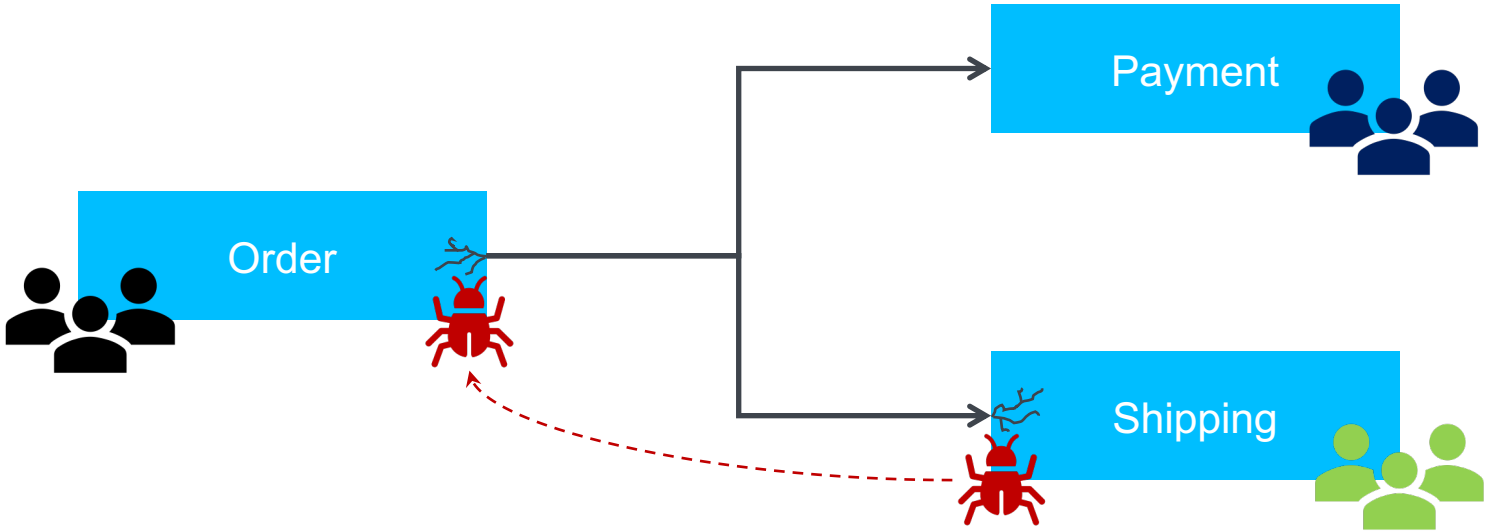
University of Stuttgart
Institute of Software Engineering (ISTE)
Software Quality and Architecture Group (SQA)

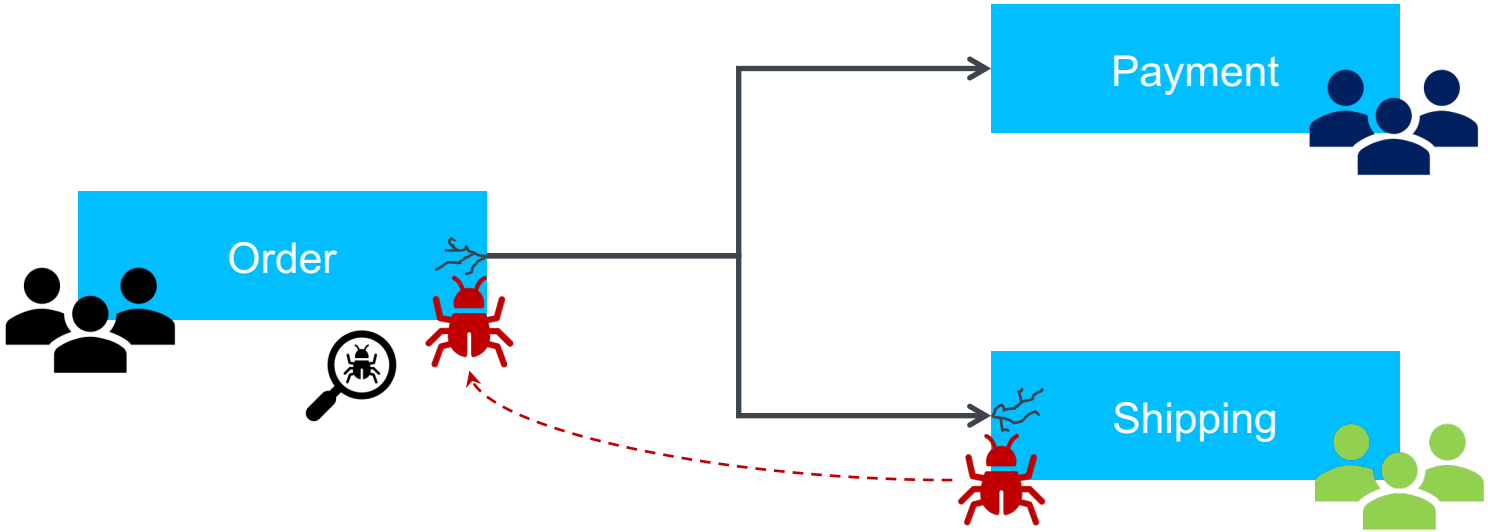
Tracing Issues that Propagate Across Microservice Boundaries

AK MSDO Treffen 2022

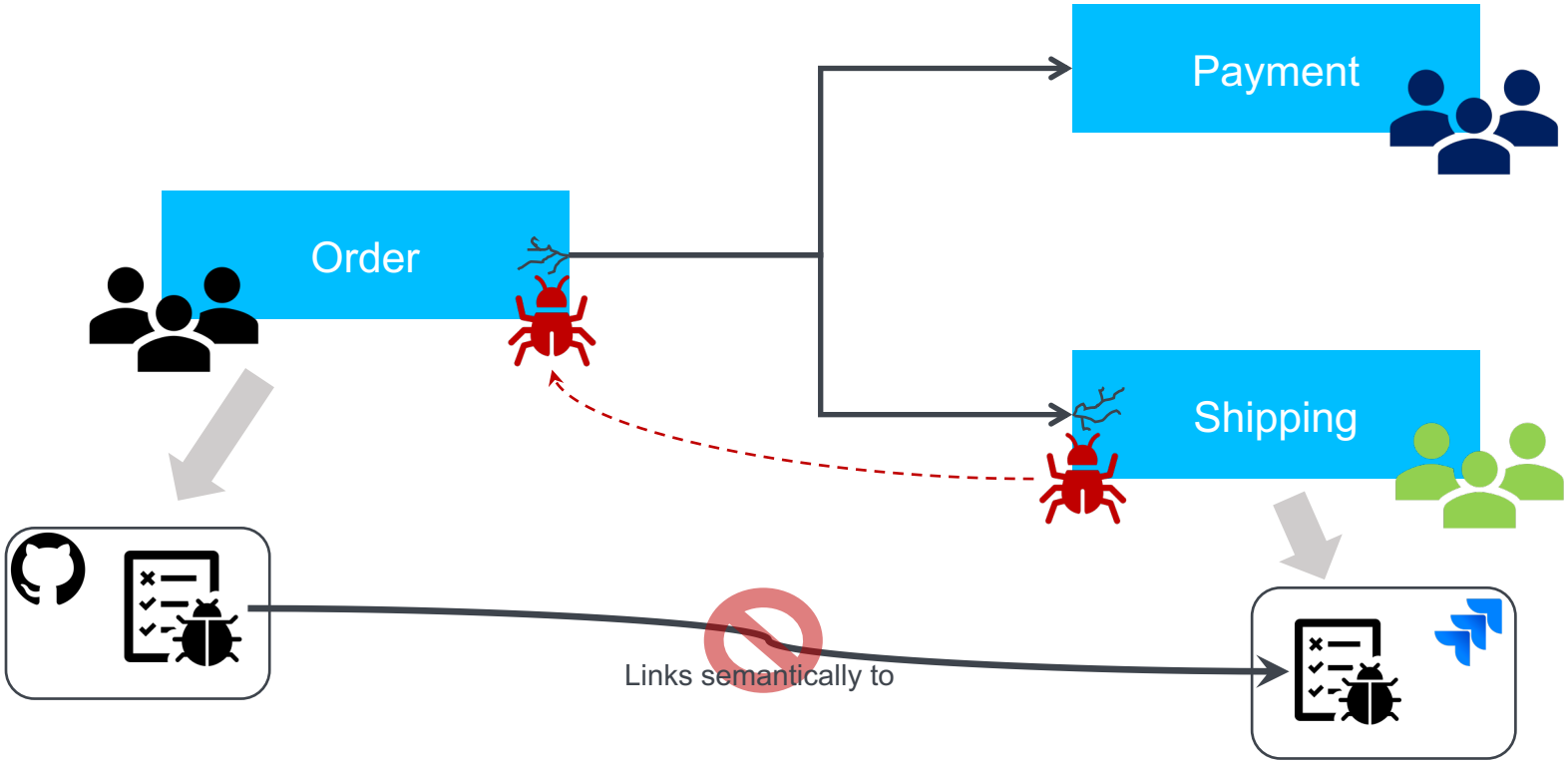
Motivation



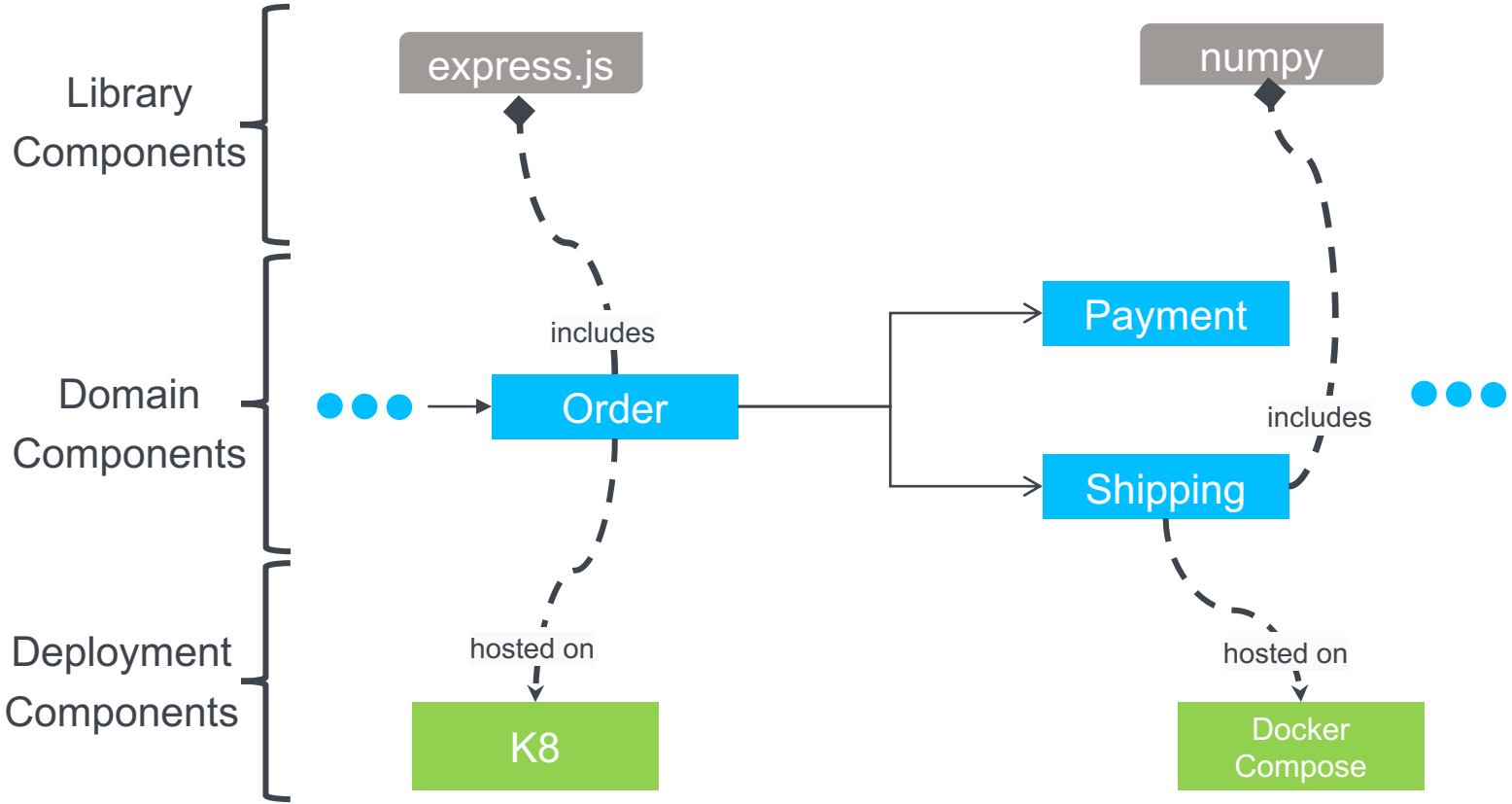
Motivation



Motivation



Motivation


Library Components
Domain Components
Deployment Components
express.js
numpy
includes
Payment
Order
includes
Shipping
hosted on
hosted on
K8
Docker Compose

Motivation

Typical examples of cross-component issues

„polyfit and eig regression tests fail after Windows 10 update to 2004“ (numpy)

Color.js[1]

„Kube DNS Latency“ (kubernetes/dns)

„kube-dns: dnsmasq intermittent connection refused“ (kubernetes/kubernetes)

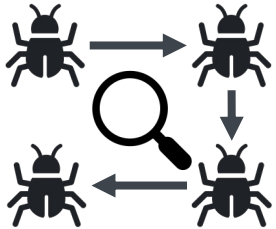
Log4j

„Pods Terminating forever due to Docker 17.09-ce Bug“ (kubernetes-retired/kube-aws)

[1] https://t3n.de/news/fakerjs-colorjs-entwickler-korrumpiert-1442982/

Problems

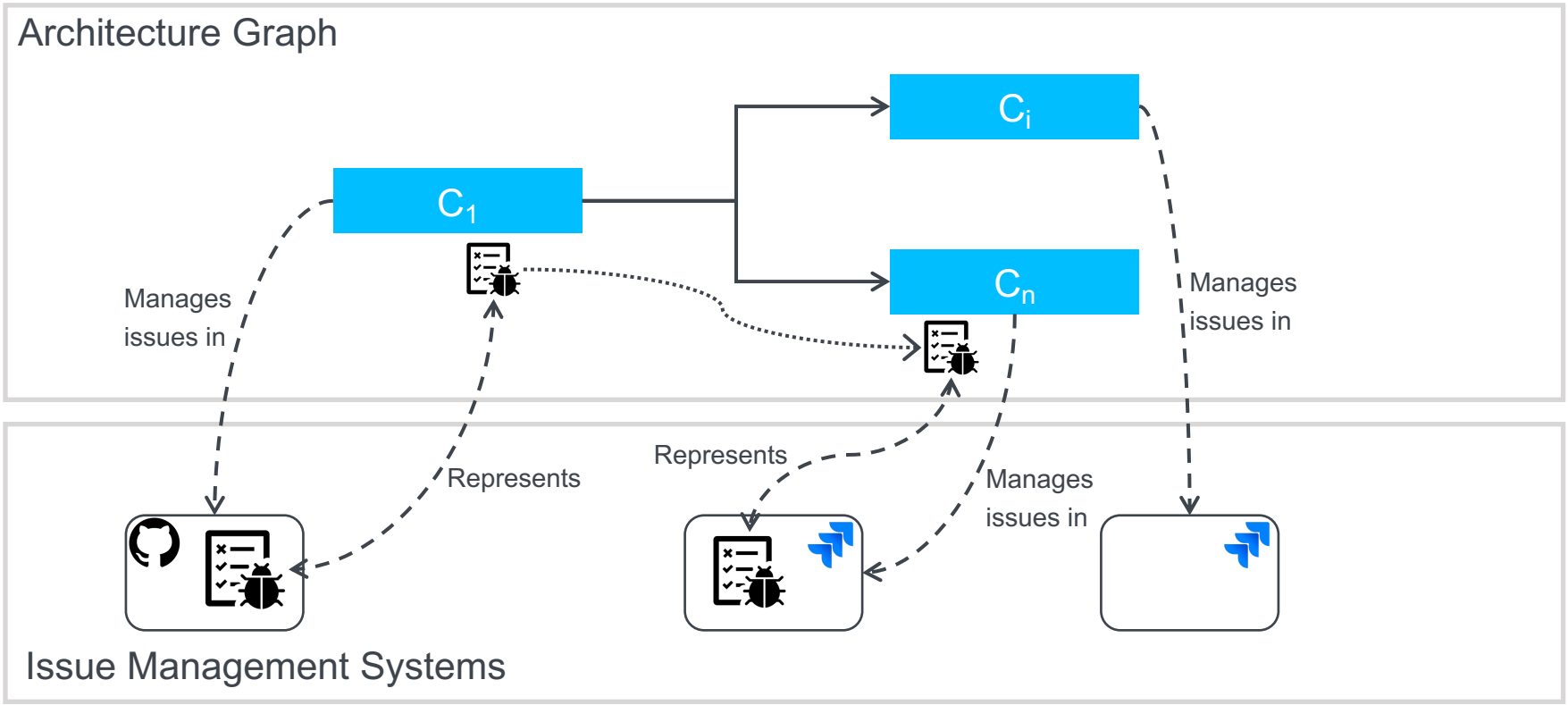
Tracing and managing issues across components



Manual identification of cross-component issues and issue relations

Different stakeholder require different component types or views

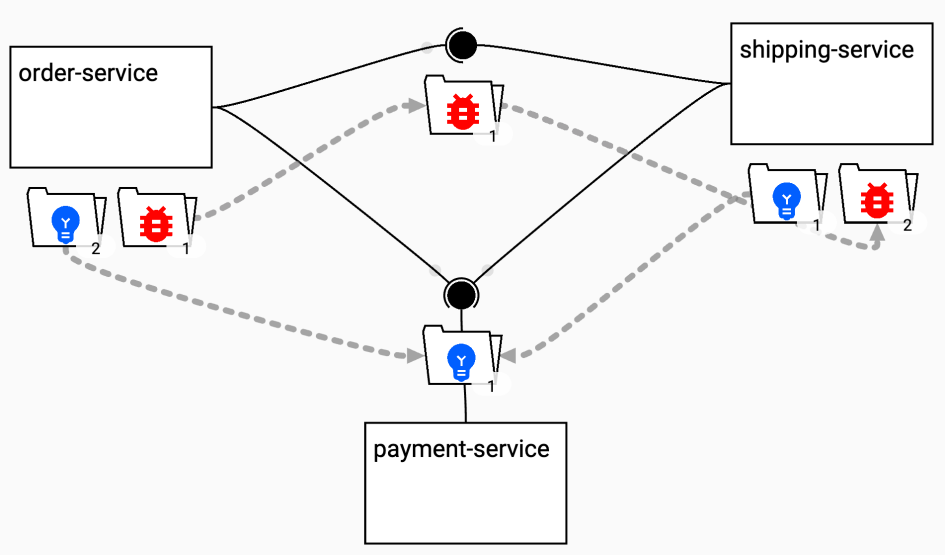
Basic Solution Idea



[1] Speth et al., “Gropius — A Tool for Managing Cross-component Issues”, 2020.
[2] Speth et al., “Cross-Component Issue Metamodel and Modelling Language”, 2021

Gropius

Gropius



- Simplicity
- Fast overview of the system’s status
- IMS-Independent
- Cognitive effective

- Simplicity
- Restricted to one component type
- Not fitted for Devs
- Not extensible
- Complex characteristics not included

Related Industry Efforts

[1] https://www.redmine.org/boards/1/topics/21939
[2] https://tinyurl.com/issues-multiple-projects
[3] https://tinyurl.com/share-issues-across-projects

Proxy Issues

Jira Multi-Project Picker

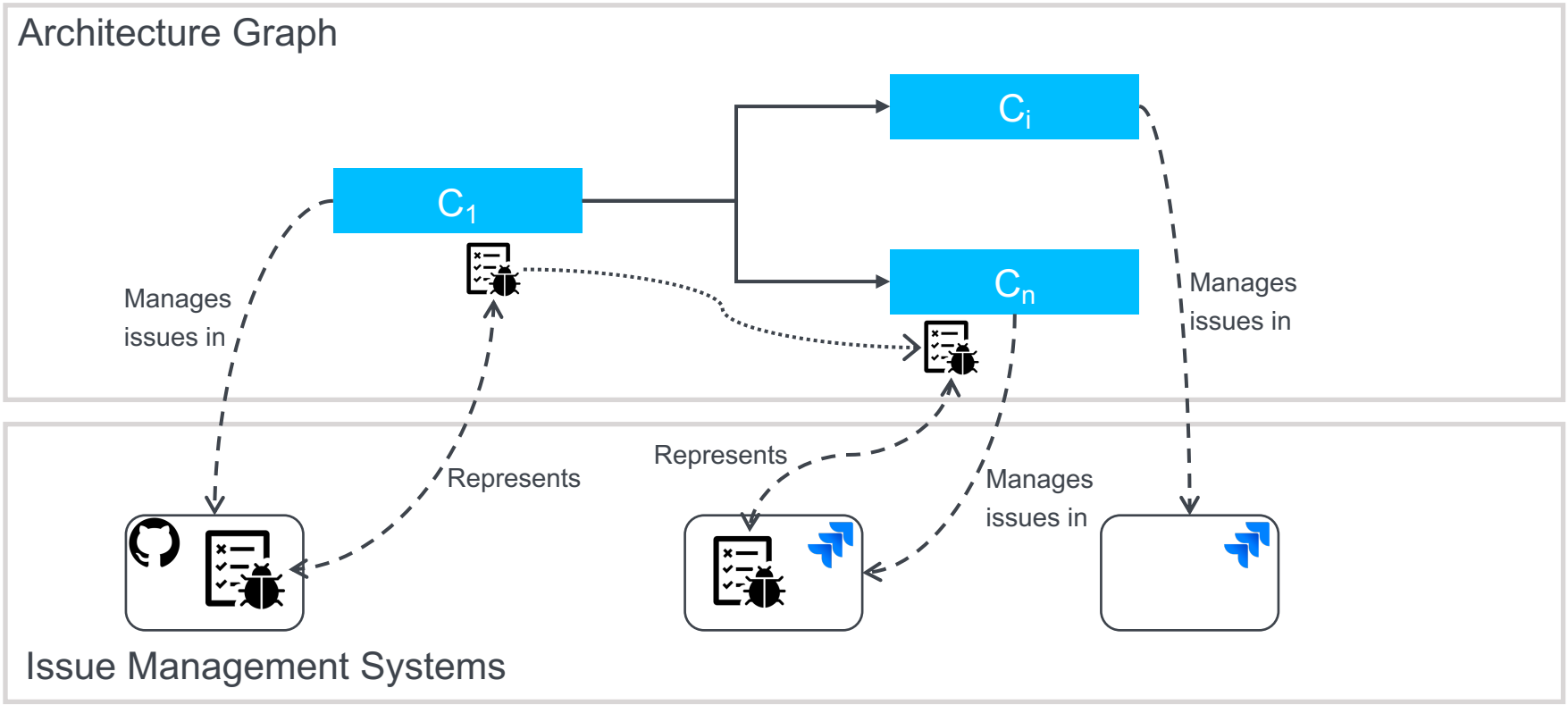
Jira Backbone Issues

URL in comments

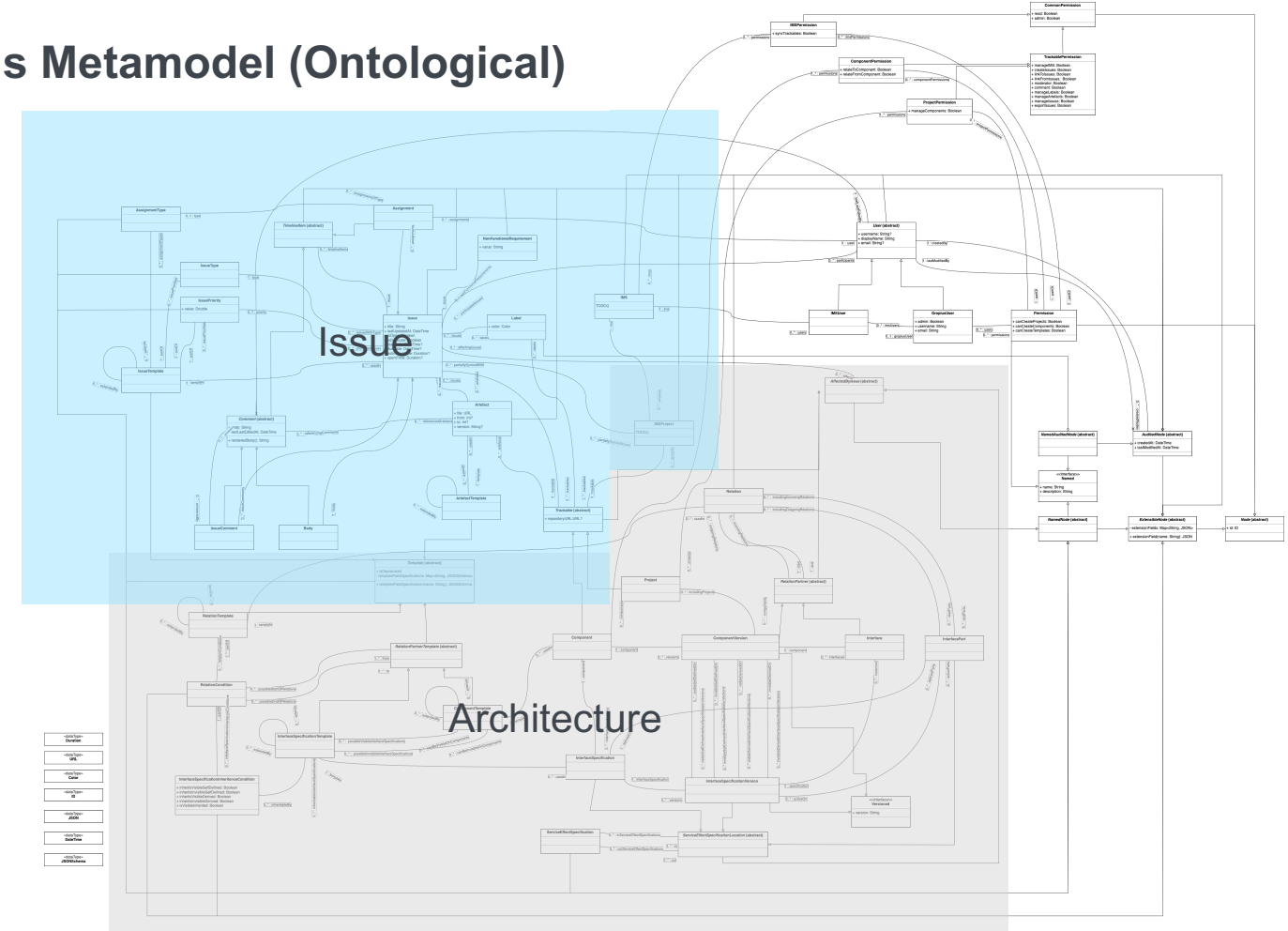
Requirements: Component Types & Characteristics

- Issues separated for type and instance
- Connect components to interfaces or components
- Different types of components
- Different types of component relations
- Versioning of component instances
- Versioning of interfaces
- Extensible for new component types and component relations
- Component-independent issues

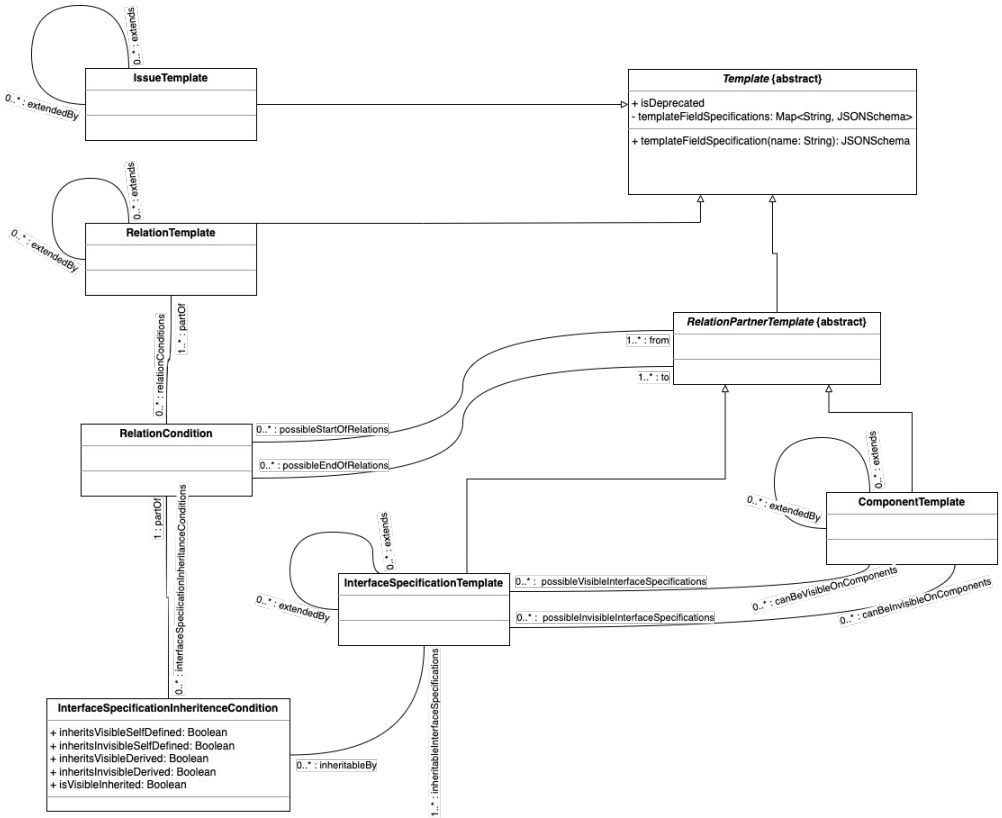
Basic Solution Idea



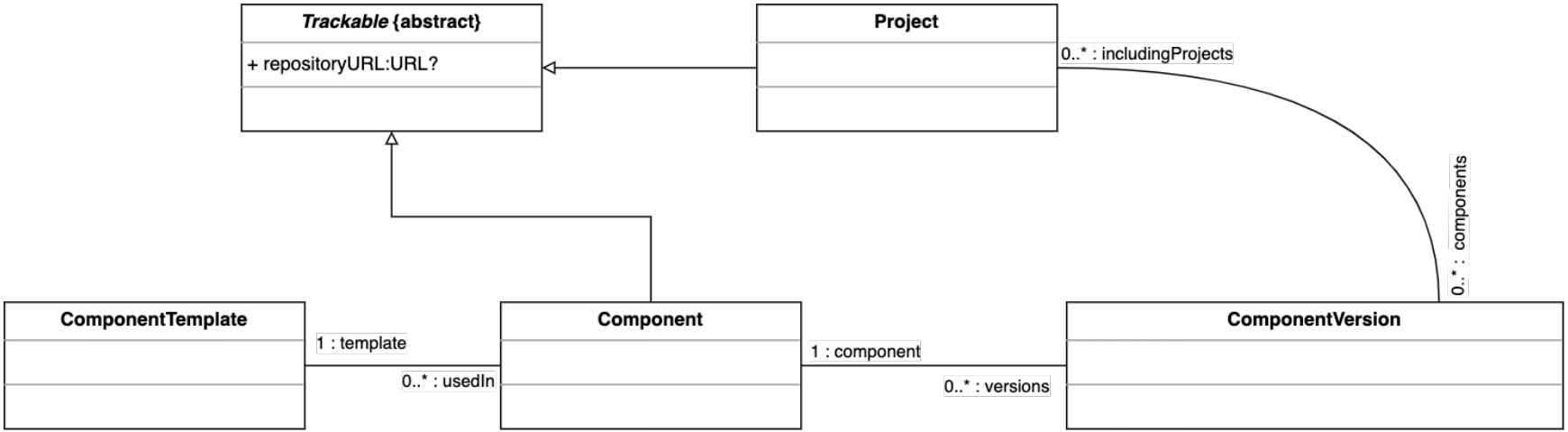
Gropius Metamodel (Ontological)



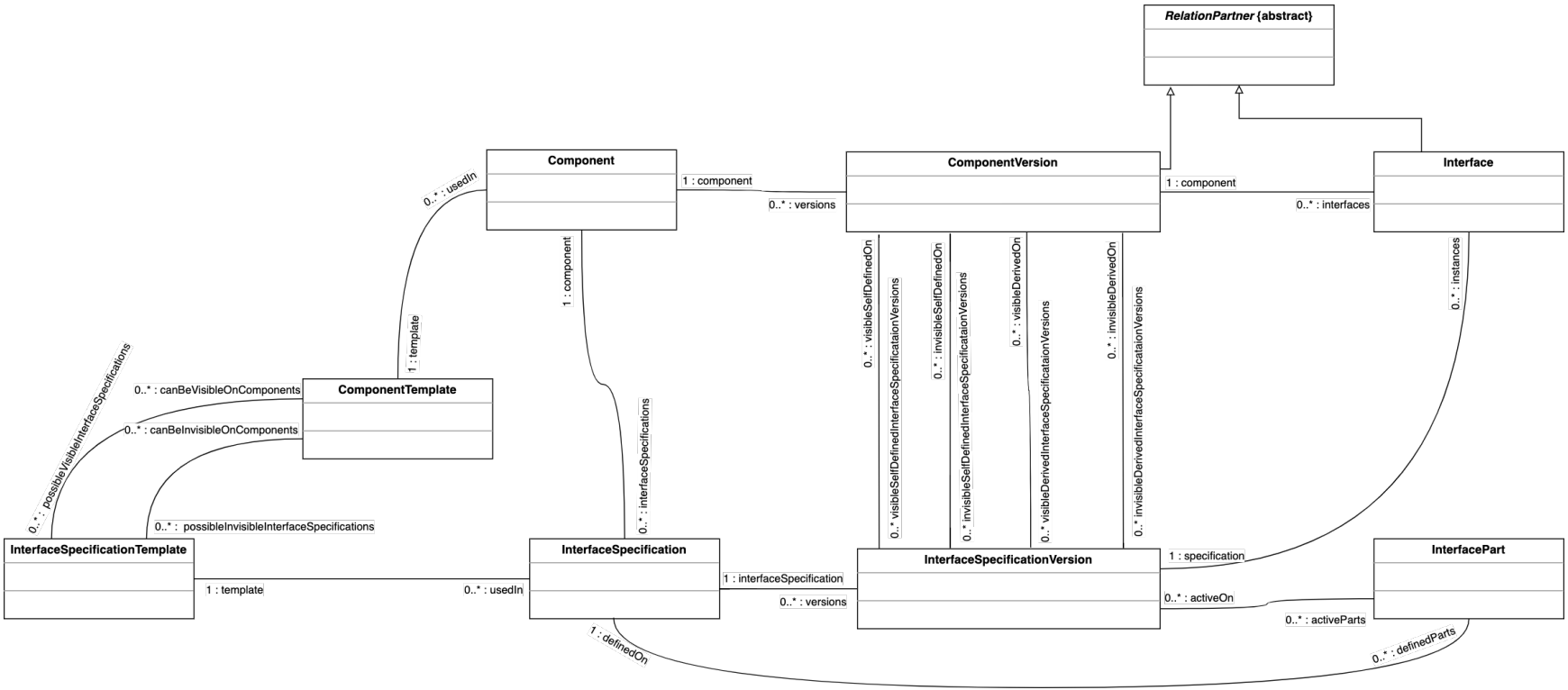
Template Mechanism



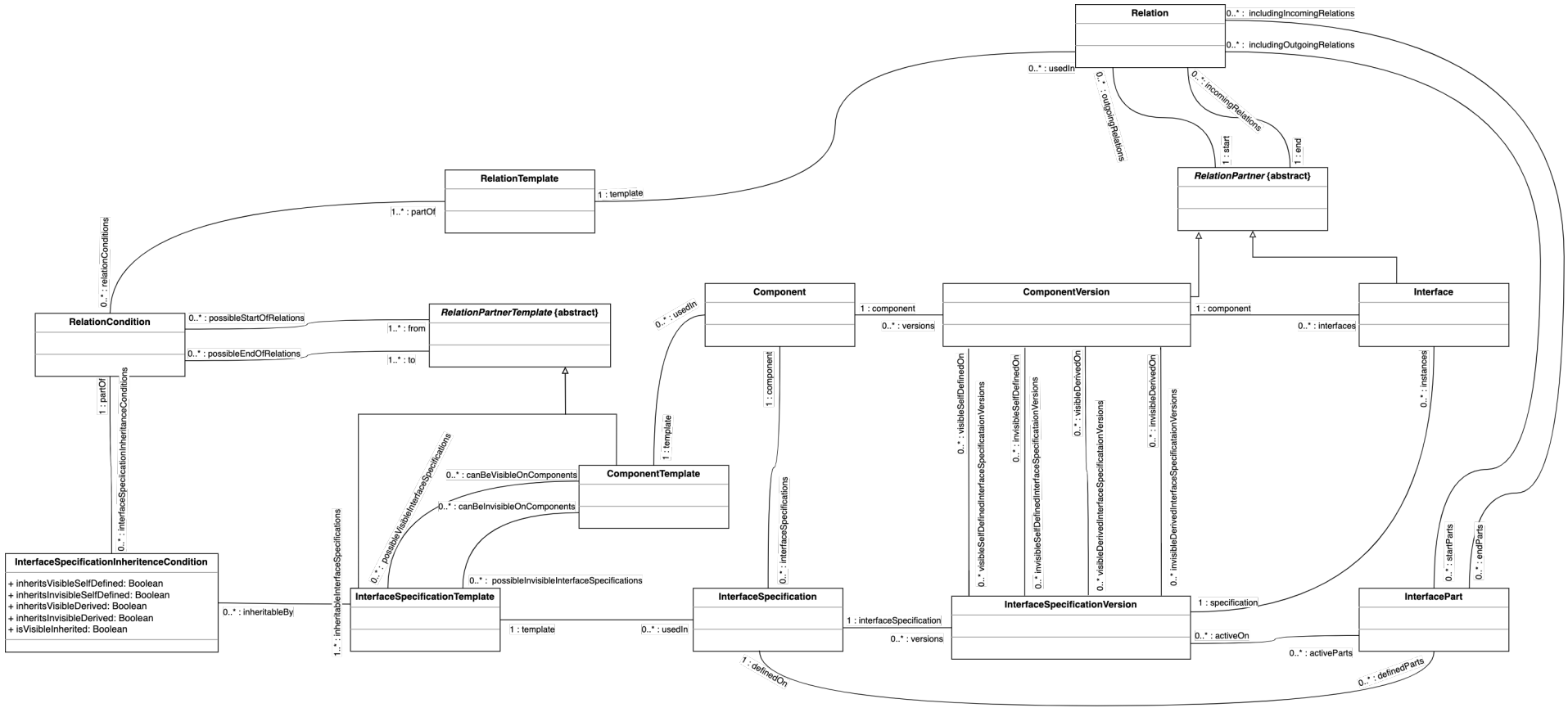
Component Template, Type, and Version



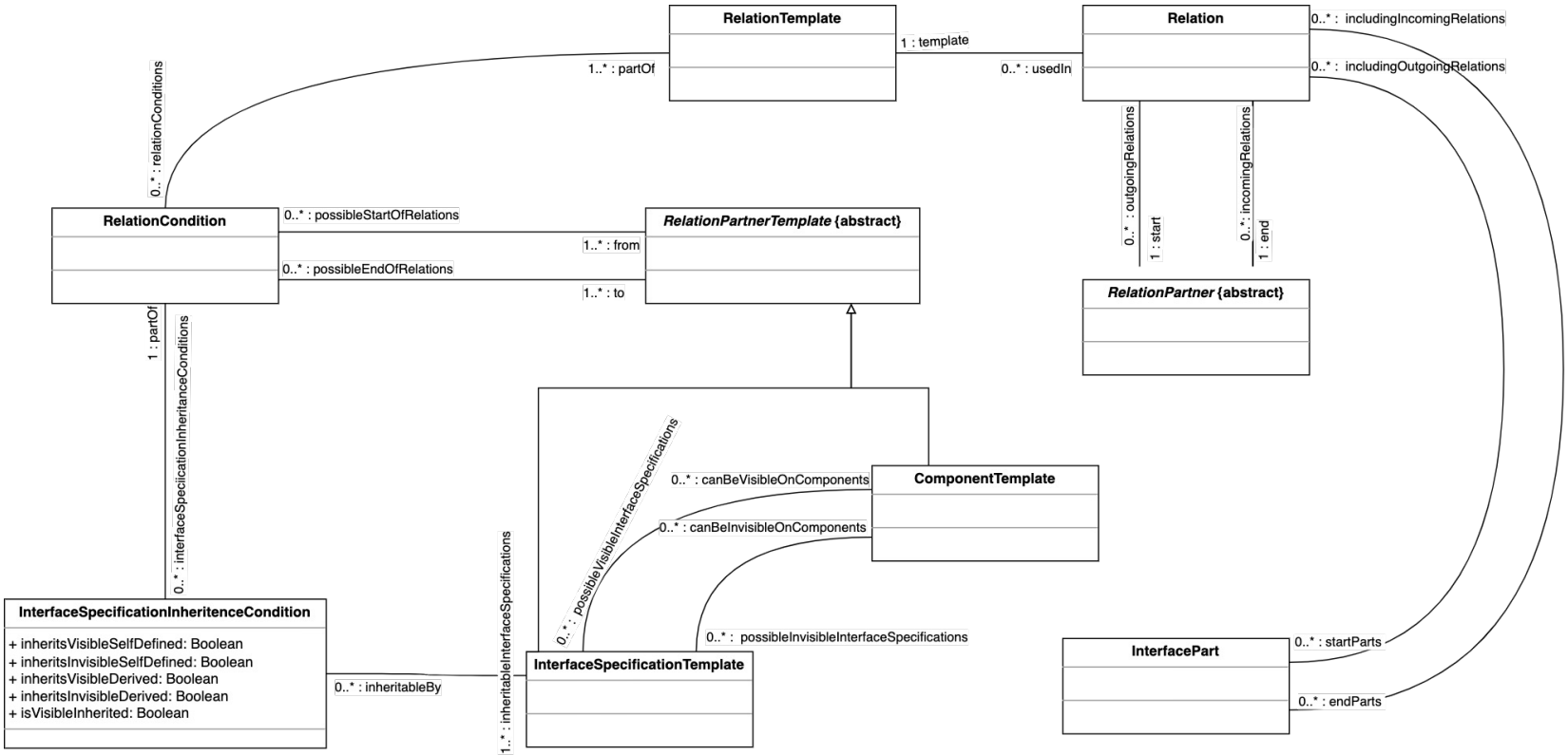
Component Interfaces



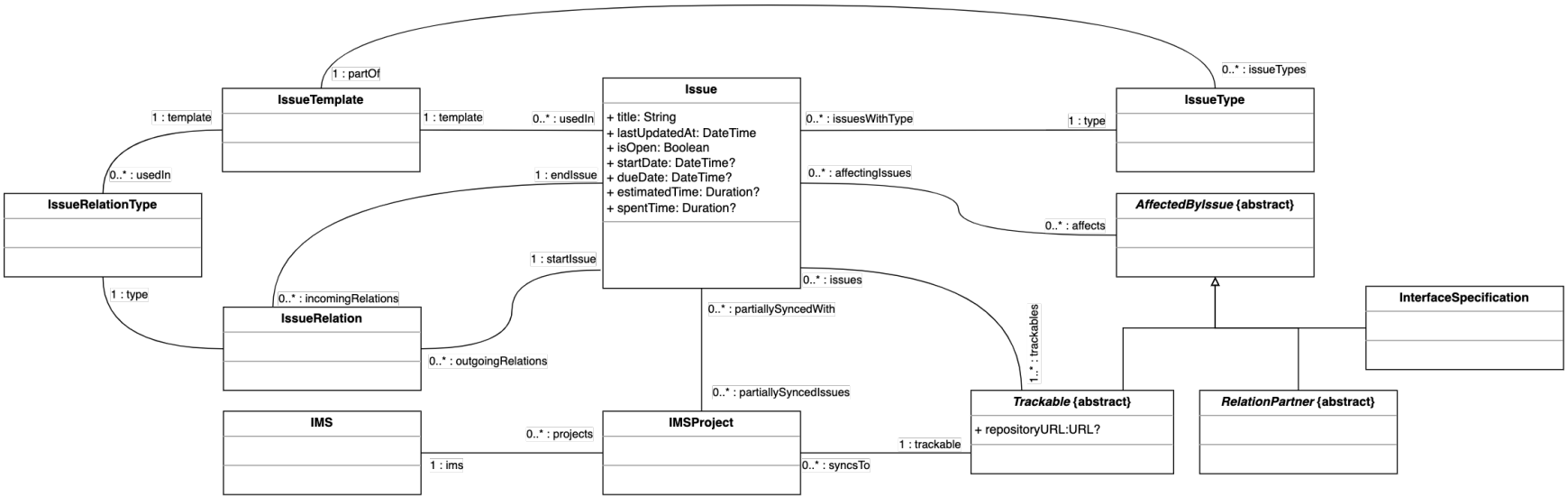
Component Relations



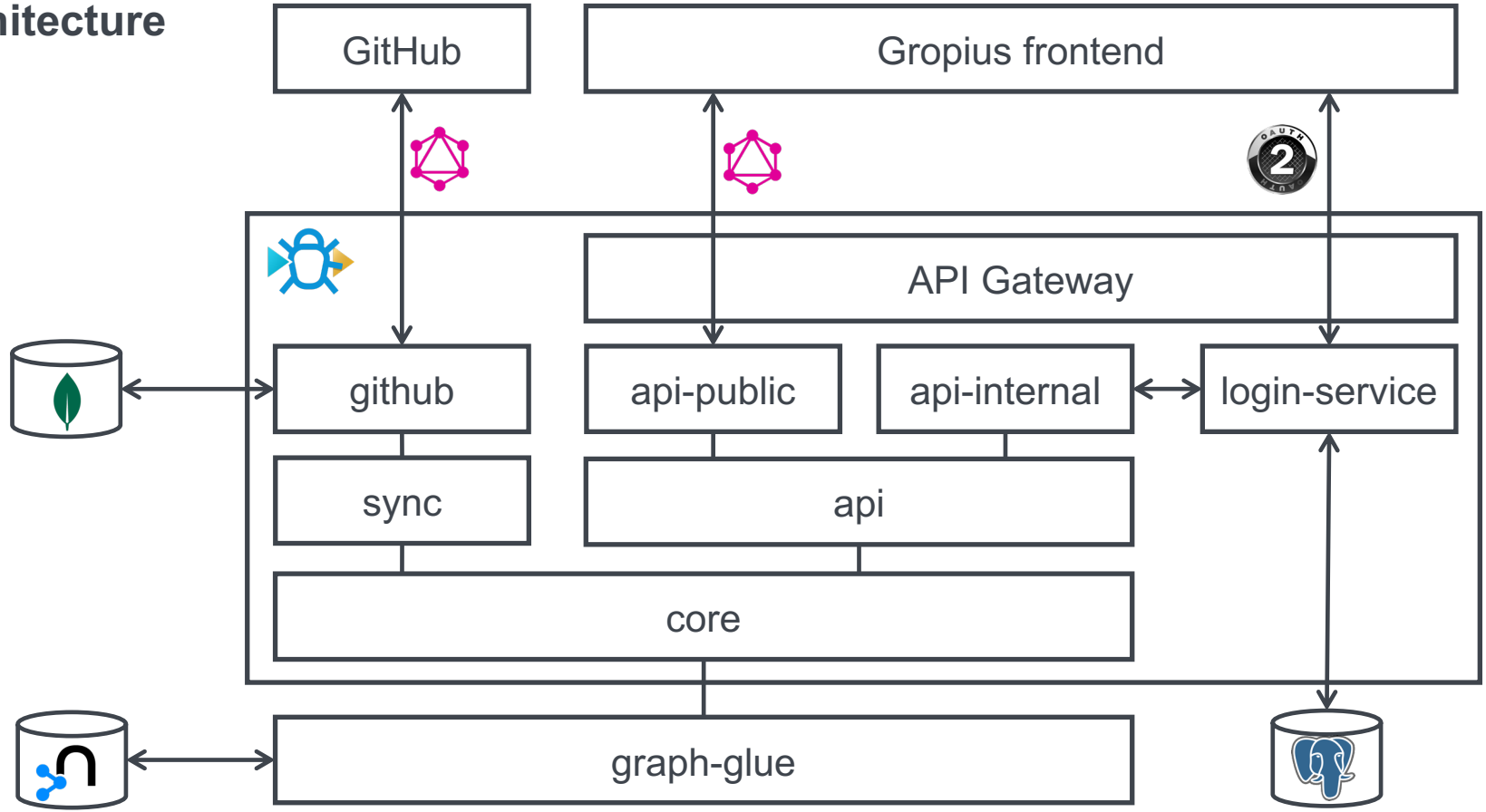
Component Relations (abstract)



Issues

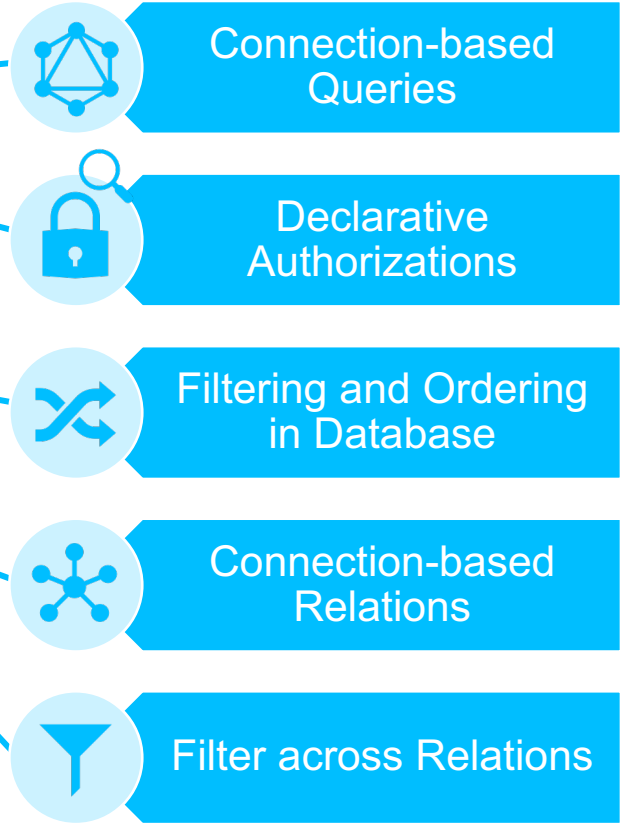

1 : partOf
0..* : issueTypes
IssueTemplate
1 : template
1 : template
0..* : usedIn
Issue
+ title: String
+ lastUpdatedAt: DateTime
+ isOpen: Boolean
+ startDate: DateTime?
+ dueDate: DateTime?
+ estimatedTime: Duration?
+ spentTime: Duration?
0..* : issuesWithType
1 : type
IssueType
0..* : usedIn
1 : endIssue
0..* : affectingIssues
IssueRelationType
AffectedByIssue {abstract}
0..* : affects
1 : startIssue
0..* : issues
1 : type
0..* : incomingRelations
IssueRelation
0..* : outgoingRelations
0..* : partiallySyncedWith
1..* : trackables
InterfaceSpecification
0..* : partiallySyncedIssues
Trackable {abstract}
+ repositoryURL:URL?
RelationPartner {abstract}
IMS
0..* : projects
IMSProject
1 : trackable
1 : ims
0..* : syncsTo

Architecture

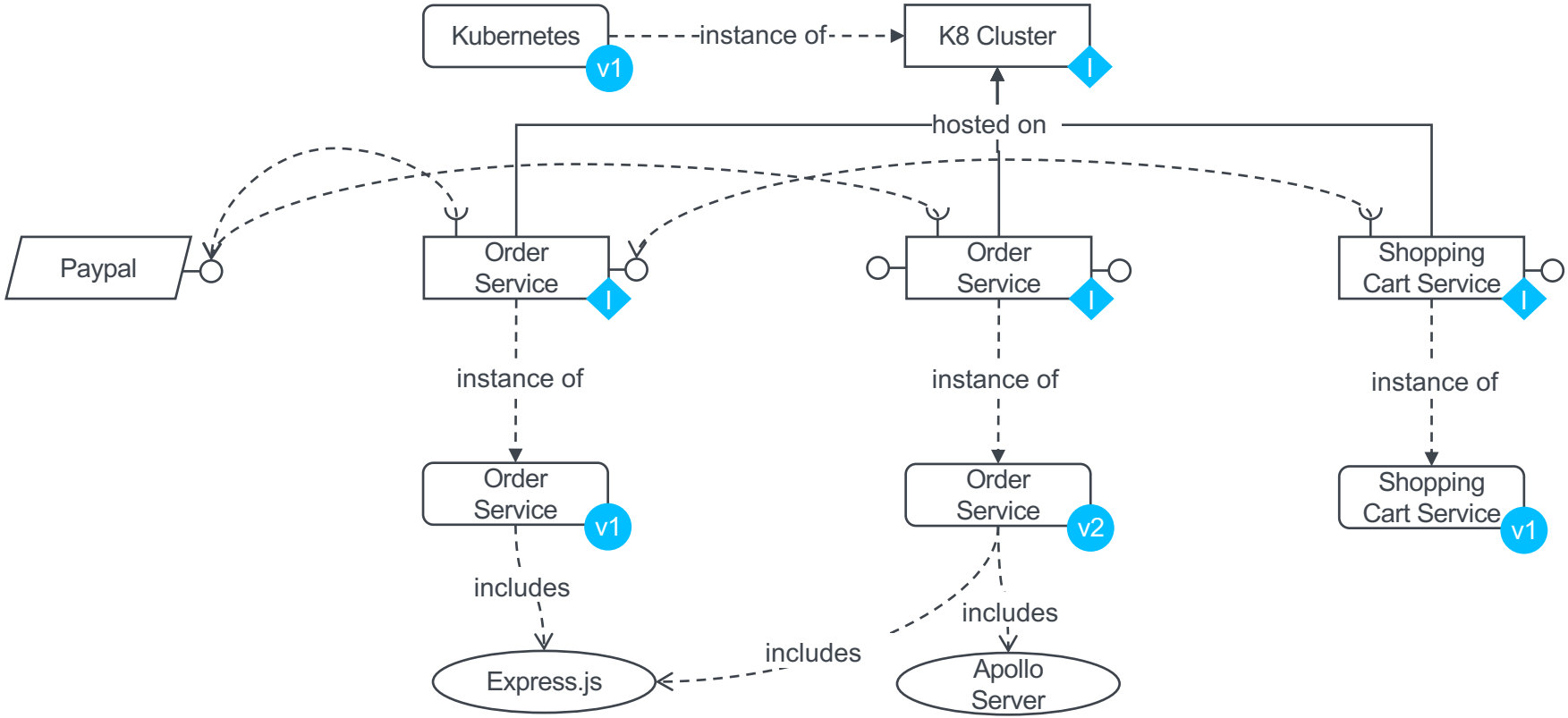


GraphGlue

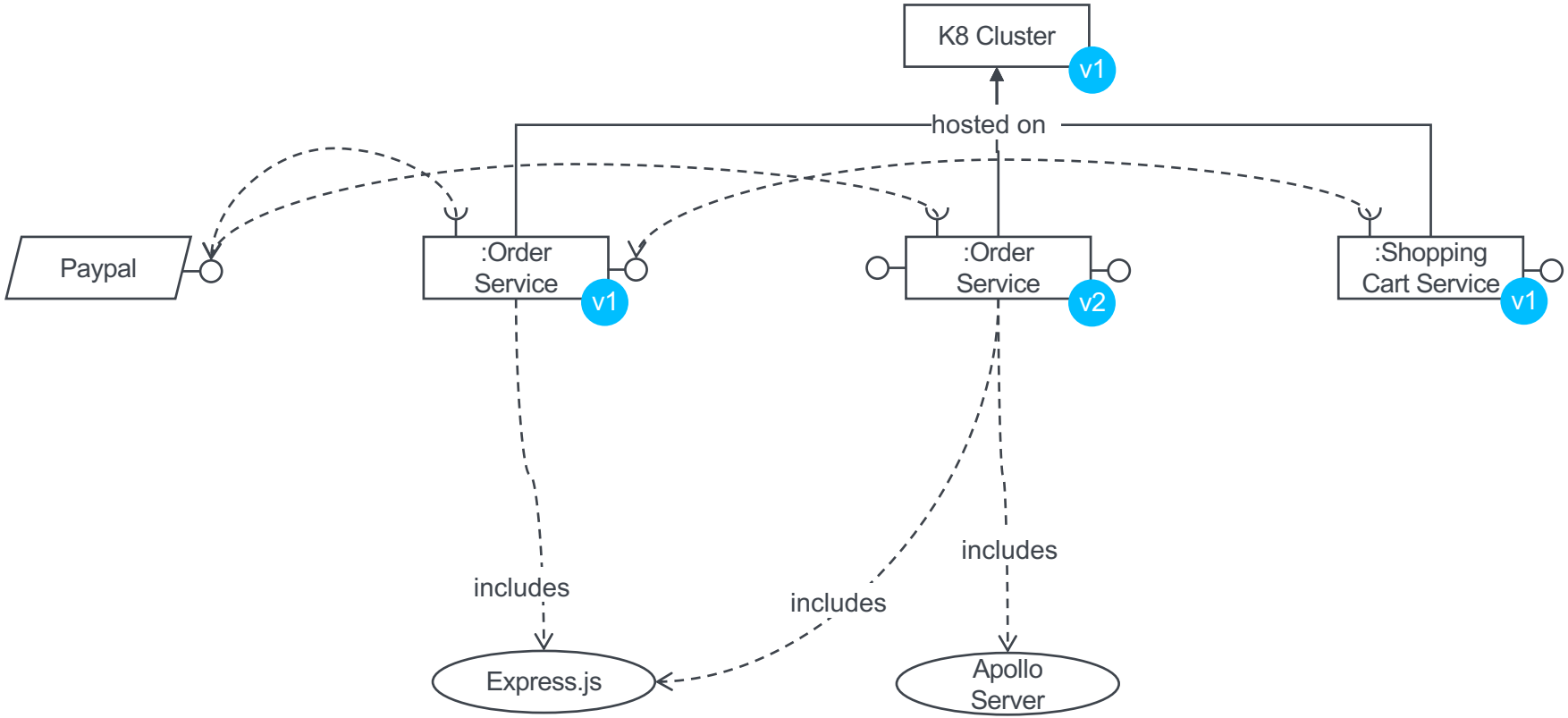
```
@DomainNode("components")
@Authorization("READ", allow = [...])
class Component(
    @OrderProperty
    @FilterProperty
    val name: String
) {

    @NodeRelationship("ISSUE", OUTGOING)
    @FilterProperty
    val issues by NodeSetProperty<Issue>()

}
```



How it COULD look like

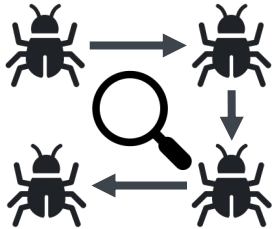


How it COULD also look like



Problems

Tracing and managing issues across components

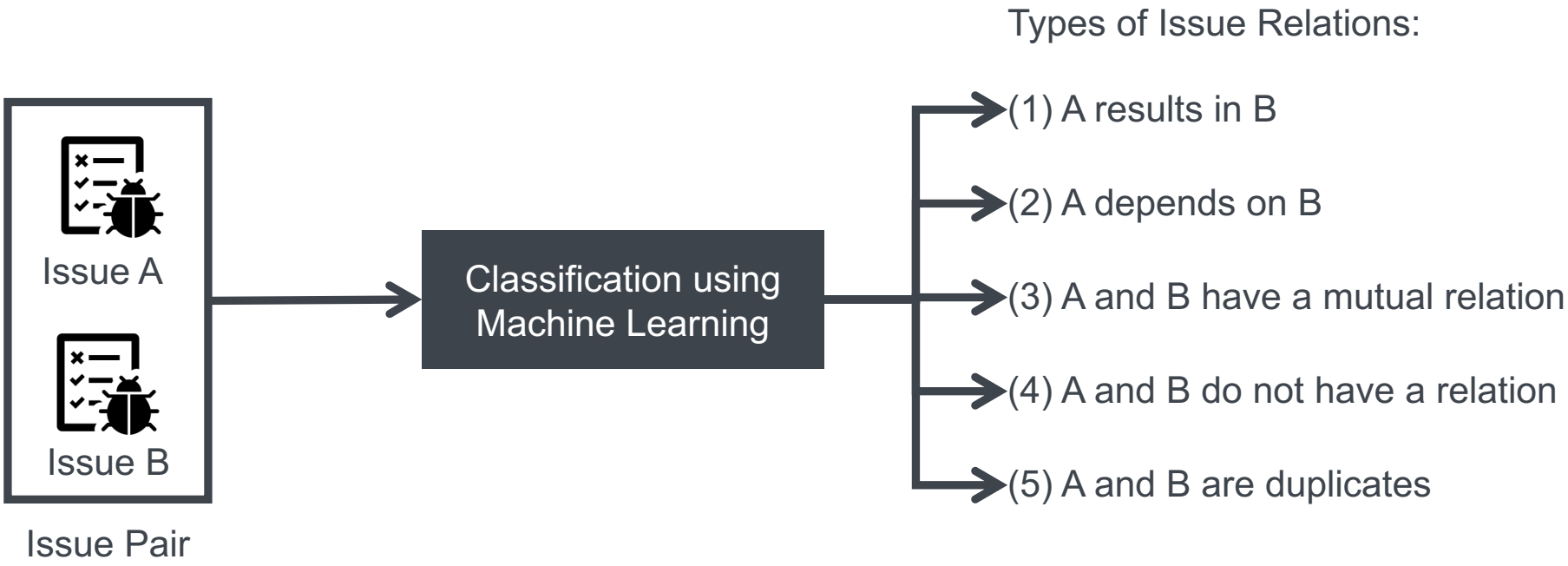


Manual identification of cross-component issues and issue relations

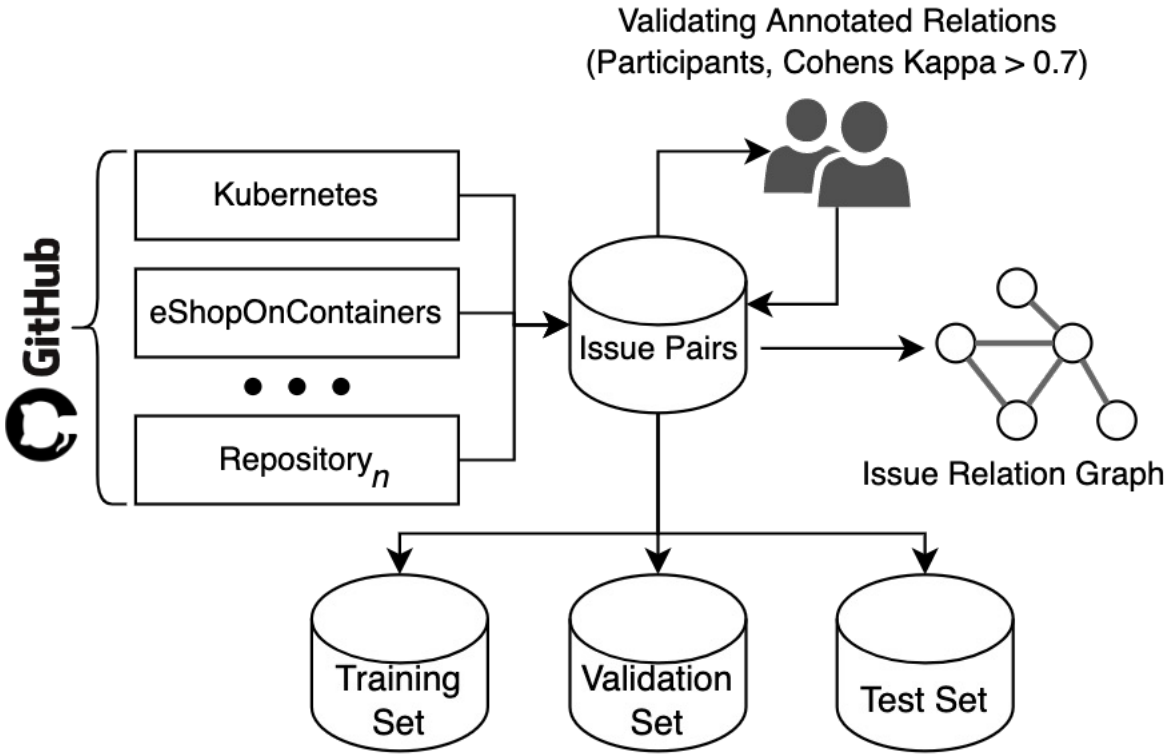
Different stakeholder require different component types or views

Automatically Identify Cross-Component Issues

Predict Relations between Issues using Classification



Data Collection



Data Annotation and Analysis

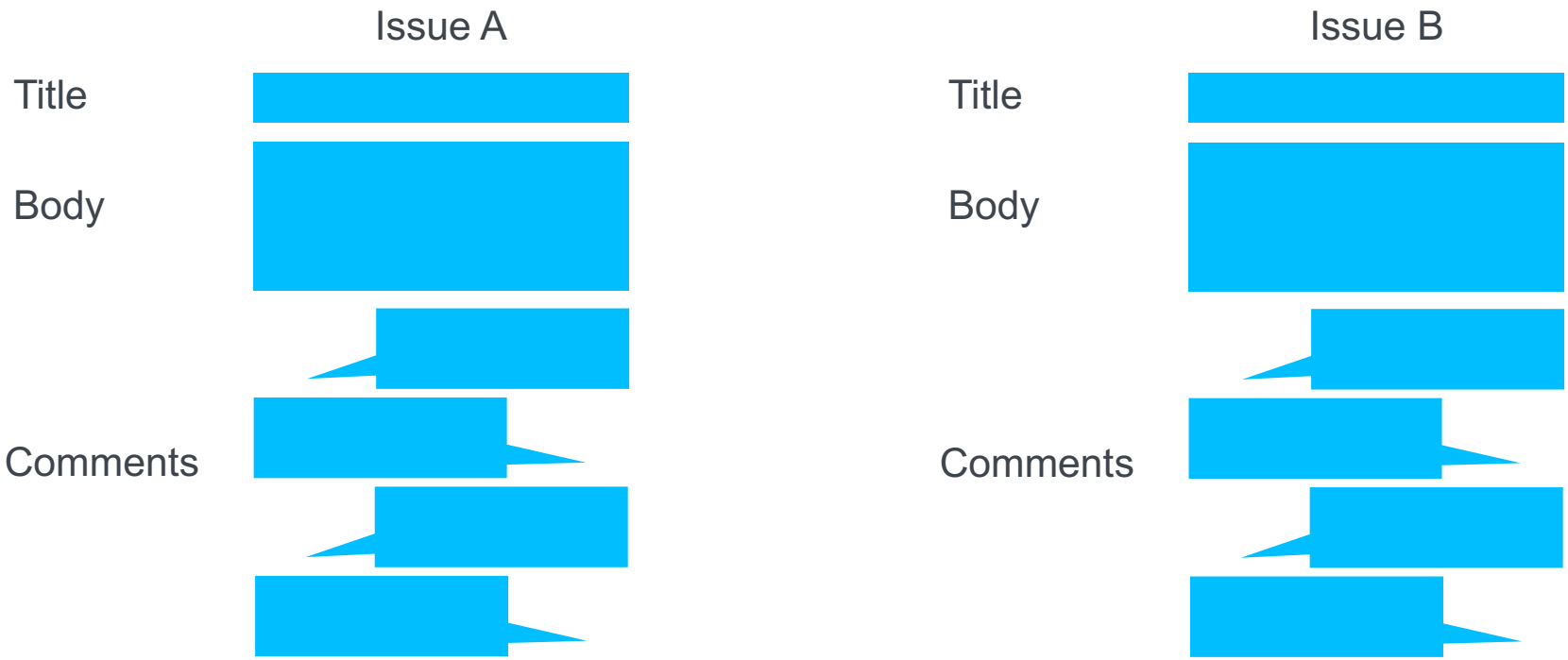
	$\rightsquigarrow$	$\leftsquigarrow$	$\leftrightsquigarrow$	*dupl*	$\not\leftrightsquigarrow$	total
collected	492	56	898	558	0	2004
generated	56	492	0	0	899	1447
train	238	238	238	238	238	1190
validation	134	134	134	134	134	670
test	176	176	176	176	176	880

The number of collected and generated issue pairs per relationship and their distribution in the datasets.

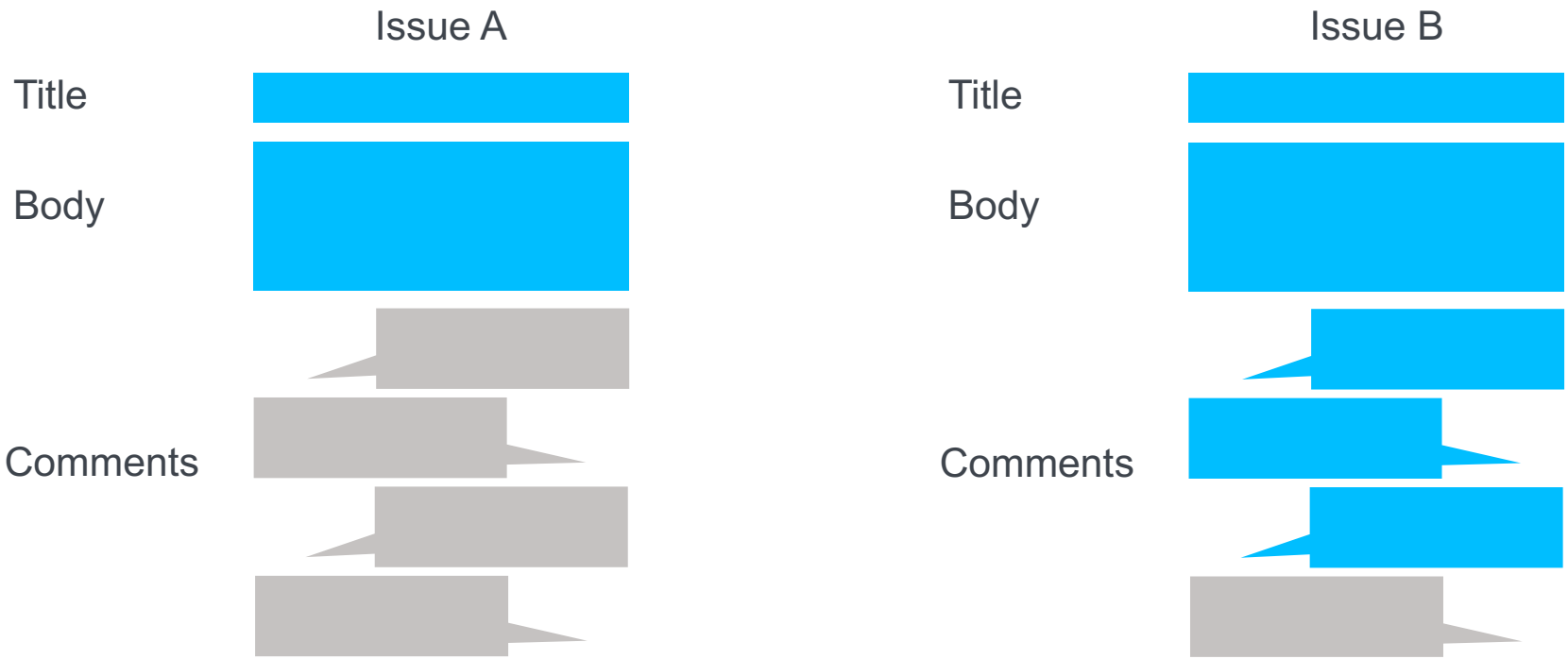
	$P2_{\rightsquigarrow}$	$P2_{\leftsquigarrow}$	$P2_{\leftrightsquigarrow}$	$P2_{dupl}$
$P1_{\rightsquigarrow}$	365	5	229	8
$P1_{\leftsquigarrow}$	10	33	26	3
$P1_{\leftrightsquigarrow}$	93	13	506	20
$P1_{dupl}$	16	5	113	519

Participant annotator relation between annotators P1 and P2.

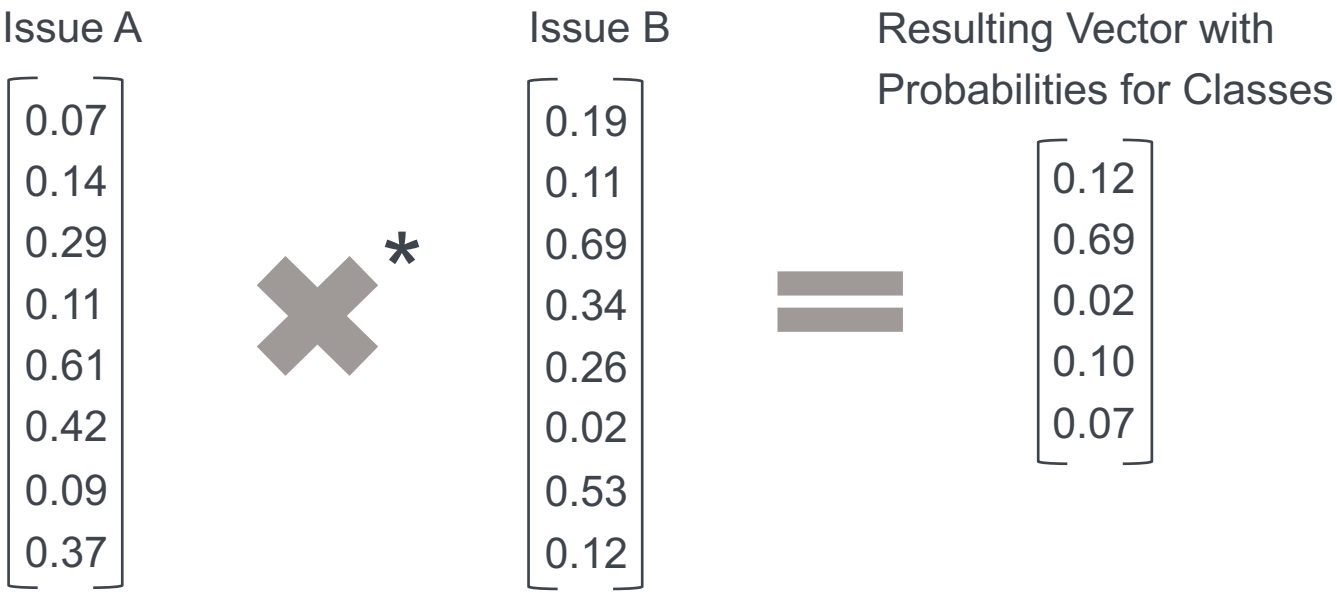
Data Creation



Identify Cross-Component Issues

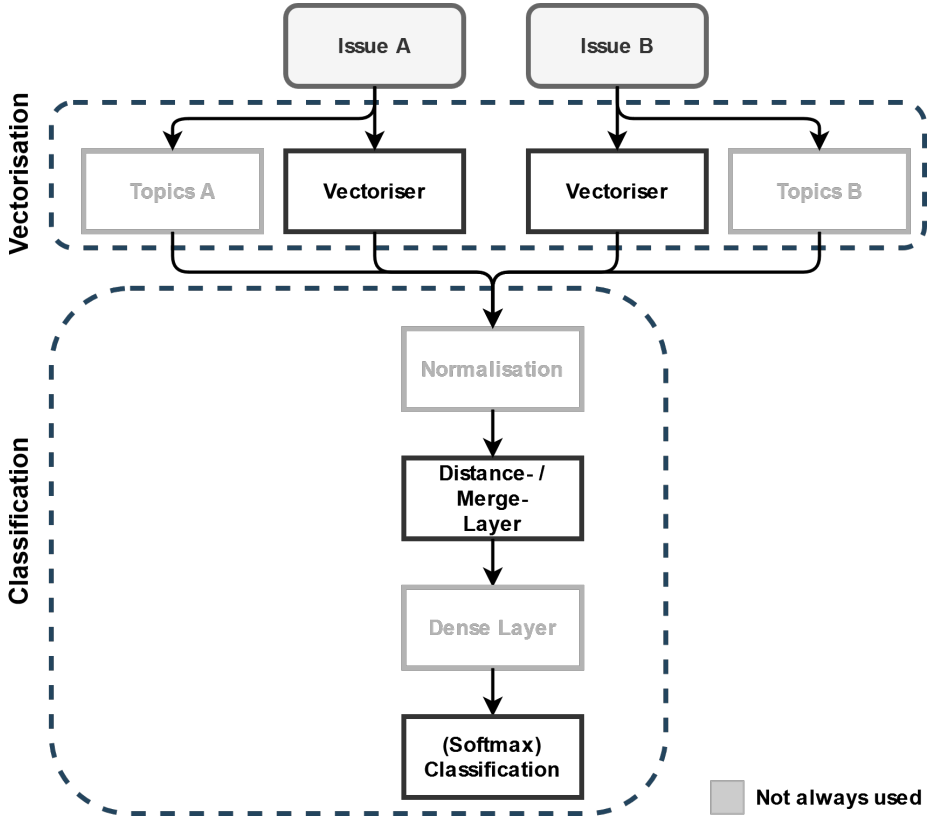


Predict Relations between Issues using Classification



* Depending on the model

Basic Classifier Architecture



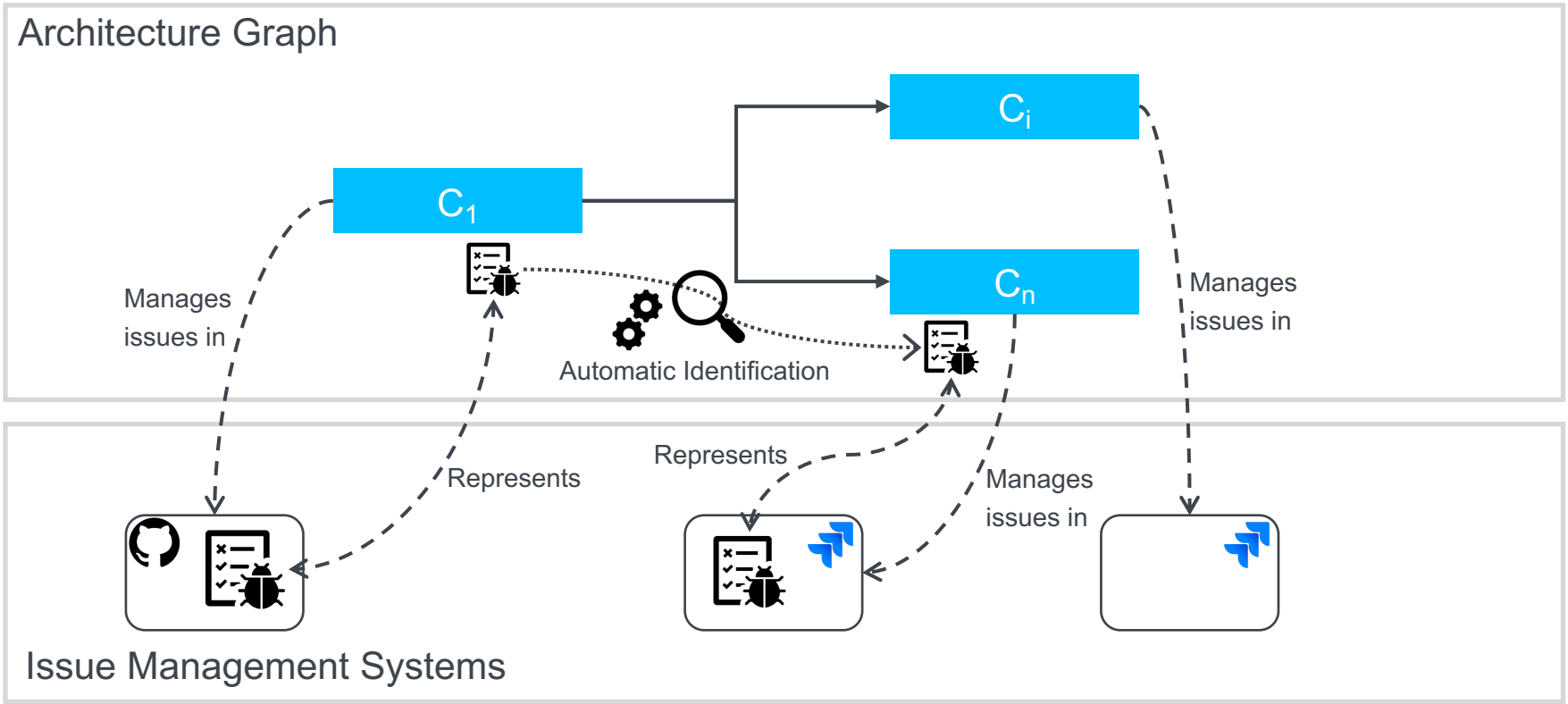
Classification Models and Results

$F_1 = 2*((P * R) / (P + R))$

				⇝			⇜			⇿			⇎			dupl			
Model	Feature	Dim.	Scope	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	MF_1
sub	tf-idf	1024	1	35	**66**	46	38	**61**	47	24	11	15	25	16	20	21	7	11	28
mult	tf-idf	1024	1,2	23	7	11	23	18	20	24	**45**	**31**	56	69	62	**33**	25	29	30
avg	count	1024	1,2	22	19	20	25	7	11	21	19	20	26	53	35	18	16	17	21
concat	count	512	1	34	23	28	45	24	31	24	22	23	34	59	43	29	34	31	31
issue_vector	count	512	1	40	30	34	40	31	35	**26**	17	20	34	57	43	27	31	29	32
cosConcat	count	8192	1	43	31	36	54	40	46	25	26	25	43	**77**	55	26	18	21	37
uniCosConcat	FastText	512	s	**51**	47	**49**	**51**	49	**50**	25	16	20	**71**	62	**66**	29	**47**	**36**	**44**
sumConcat	count	8192	s	47	26	34	49	38	43	**26**	19	22	39	75	52	27	28	28	35
topic	count	1024	1,2	34	38	36	44	26	33	23	17	20	39	66	49	21	18	19	31

Precision, recall and F_1 scores for the different models and issue relations. MF_1 denotes macro F_1. The scope indicates either unigrams (1), bigrams (2), or a sentence-level embedding (s).

Summary


Architecture Graph
C_1
C_i
C_n
Manages issues in
Automatic Identification
Manages issues in
Represents
Represents
Manages issues in
Issue Management Systems

University of Stuttgart
Institute of Software Engineering (ISTE)
Software Quality and Architecture Group (SQA)

@spethso

/in/sandro-speth

Thank you!

Sandro Speth

e-mail sandro.speth@iste.uni-stuttgart.de
phone +49 (0) 711 685-61693
www. iste.uni-stuttgart.de/institute/team/Speth-00002/

University of Stuttgart
Institute of Software Engineering,
Software Quality and Architecture Group
Universitätsstraße 38,
70569 Stuttgart
Room 1.336