MetaLib – A Chrestomathy of DSL Implementations

Simon Schauss[1], **Ralf Lämmel**[1,2], Johannes Härtel[1], Marcel Heinz[1], Kevin Klein[1], Lukas Härtel[1] and Thorsten Berger[3]

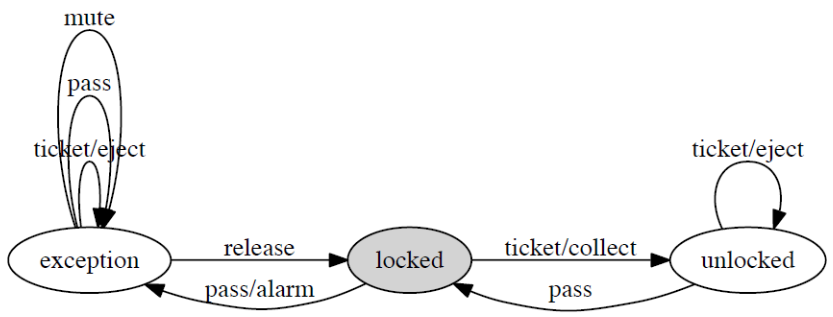
[1] University of Koblenz-Landau
[2] Facebook
[3] Chalmers | University of Gothenburg

http://www.softlang.org/metalib

How to implement an FSM Language?

```
turnstile = fsm()
        .addState("locked")
                .addTransition("ticket", "collect", "unlocked")
                .addTransition("pass", "alarm", "exception")
        .addState("unlocked")
                .addTransition("ticket", "eject", "unlocked")
                .addTransition("pass", null, "locked")
        .addState("exception")
                .addTransition("ticket", "eject", "exception")
                .addTransition("pass", null, "exception")
                .addTransition("mute", null, "exception")
                .addTransition("release", null, "locked");
```

```
initial state locked {
  ticket/collect —> unlocked;
  pass/alarm —> exception;
}
state unlocked {
  ticket/eject;
  pass —> locked;
}
state exception { ... }
```



```
fsm {

 initial state stateA {
                 eventI / actionI -> stateB ;
                }
 <cell
   <no initial> state stateB {                          >
                          eventII / actionII -> stateA ;
                         }
}
```

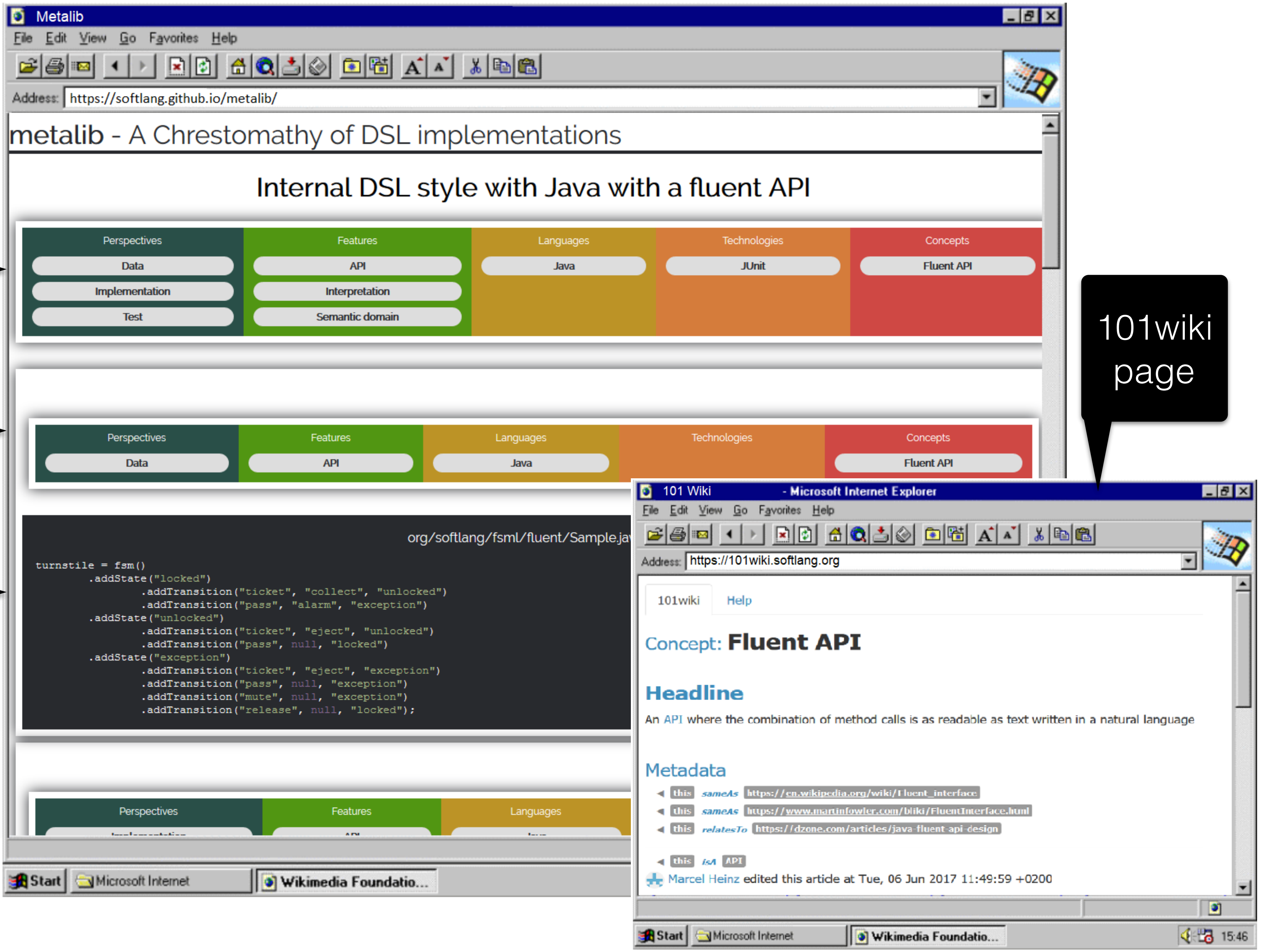
MetaLib — a chrestomathy for learning and teaching

Aggregated annotations

Snippet annotations

Snippet

101wiki page



metalib - A Chrestomathy of DSL implementations

Internal DSL style with Java with a fluent API

org/softlang/fsml/fluent/Sample.ja

```
turnstile = fsm()
        .addState("locked")
                .addTransition("ticket", "collect", "unlocked")
                .addTransition("pass", "alarm", "exception")
        .addState("unlocked")
                .addTransition("ticket", "eject", "unlocked")
                .addTransition("pass", null, "locked")
        .addState("exception")
                .addTransition("ticket", "eject", "exception")
                .addTransition("pass", null, "exception")
                .addTransition("mute", null, "exception")
                .addTransition("release", null, "locked");
```

Concept: **Fluent API**

Headline

An API where the combination of method calls is as readable as text written in a natural language

Metadata

- this *sameAs* https://en.wikipedia.org/wiki/Fluent_interface
- this *sameAs* https://www.martinfowler.com/bliki/FluentInterface.html
- this *relatesTo* https://dzone.com/articles/java-fluent-api-design
- this *isA* API

Marcel Heinz edited this article at Tue, 06 Jun 2017 11:49:59 +0200

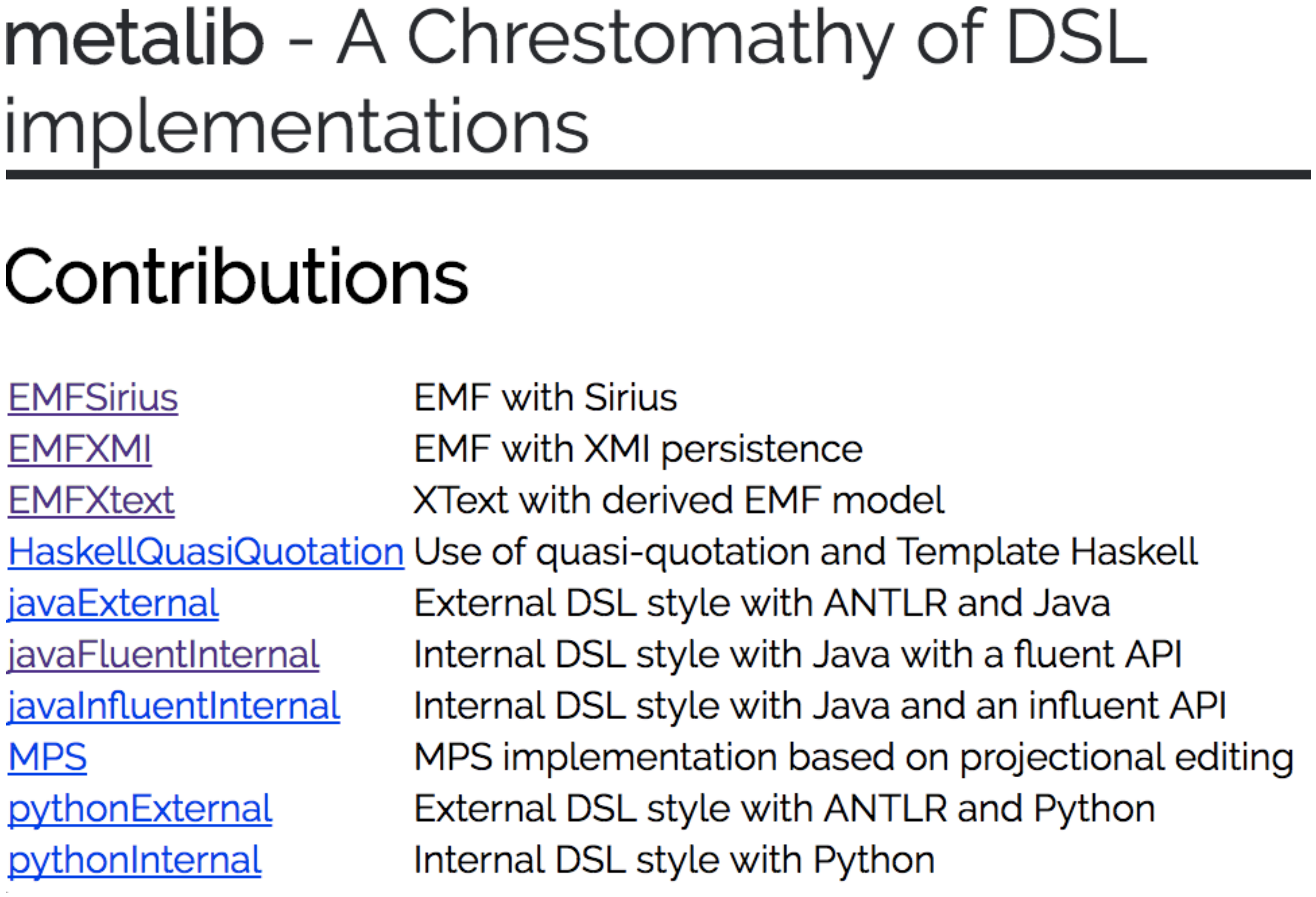
‘What’

What are the subjects of MetaLib?

We present … a software chrestomathy … for implementations of a domain-specific language (DSL).

4

http://www.softlang.org/metalib

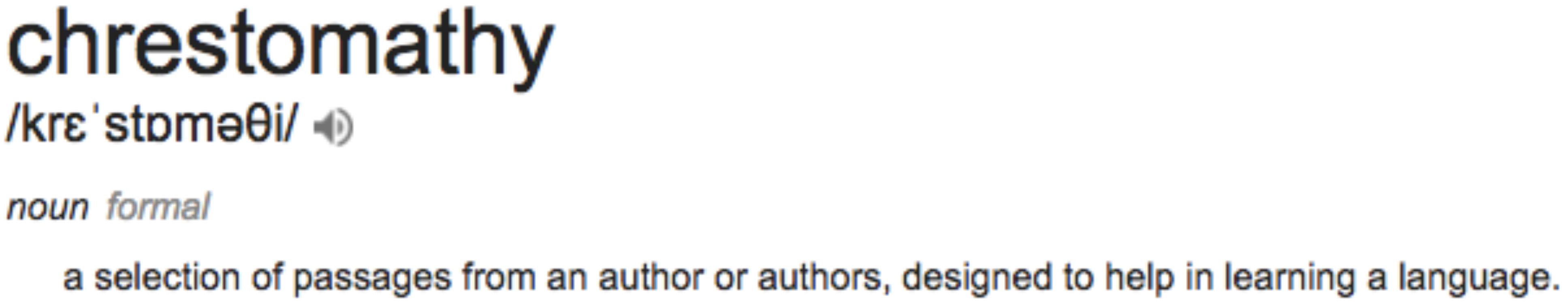


metalib - A Chrestomathy of DSL implementations

Contributions

EMFSirius	EMF with Sirius
EMFXMI	EMF with XMI persistence
EMFXtext	XText with derived EMF model
HaskellQuasiQuotation	Use of quasi-quotation and Template Haskell
javaExternal	External DSL style with ANTLR and Java
javaFluentInternal	Internal DSL style with Java with a fluent API
javaInfluentInternal	Internal DSL style with Java and an influent API
MPS	MPS implementation based on projectional editing
pythonExternal	External DSL style with ANTLR and Python
pythonInternal	Internal DSL style with Python

What’s a software chrestomathy?



chrestomathy

/krɛˈstɒməθi/

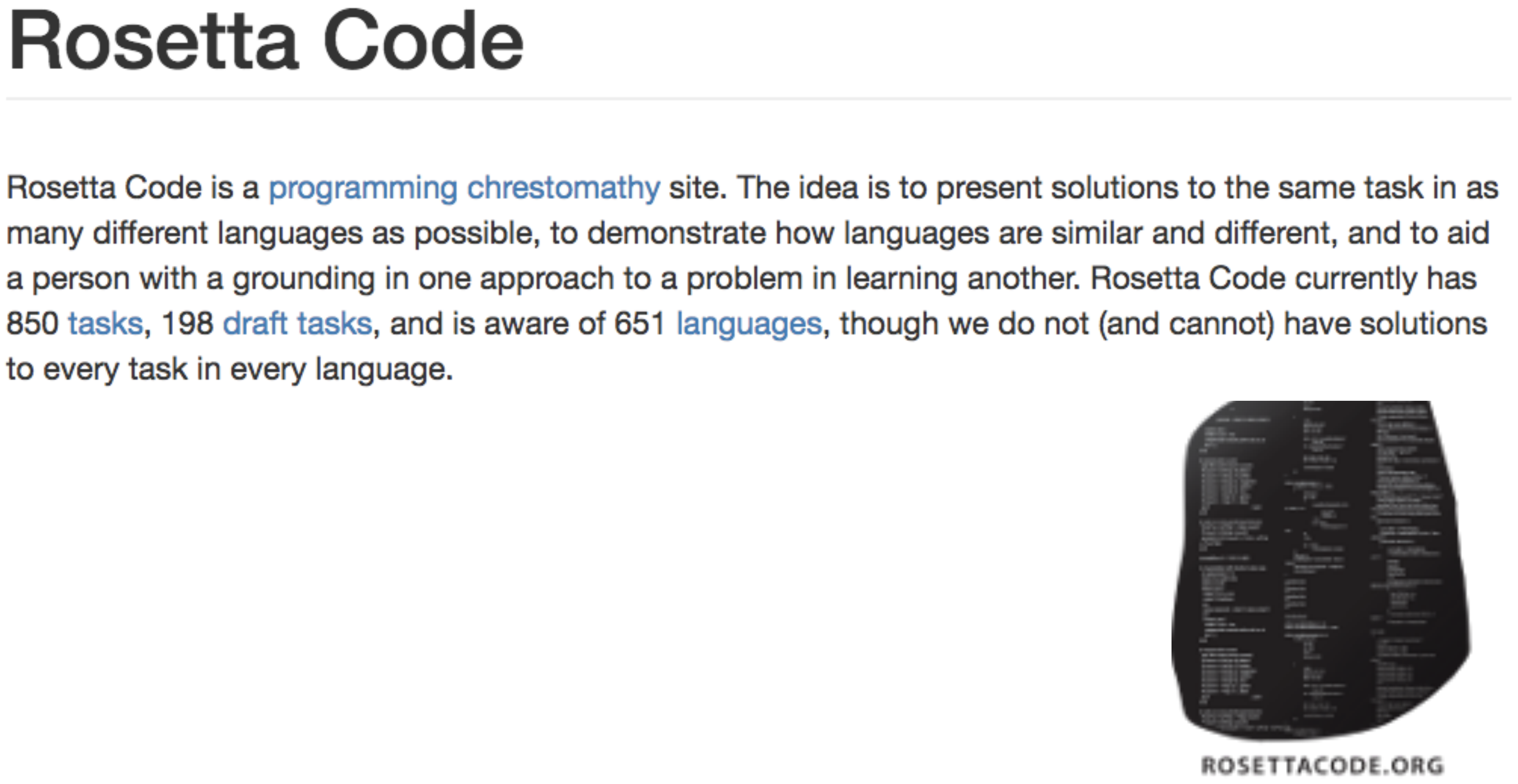
noun *formal*

a selection of passages from an author or authors, designed to help in learning a language.

[Google]

Another example of a software chrestomathy

http://rosettacode.org/wiki/Rosetta_Code



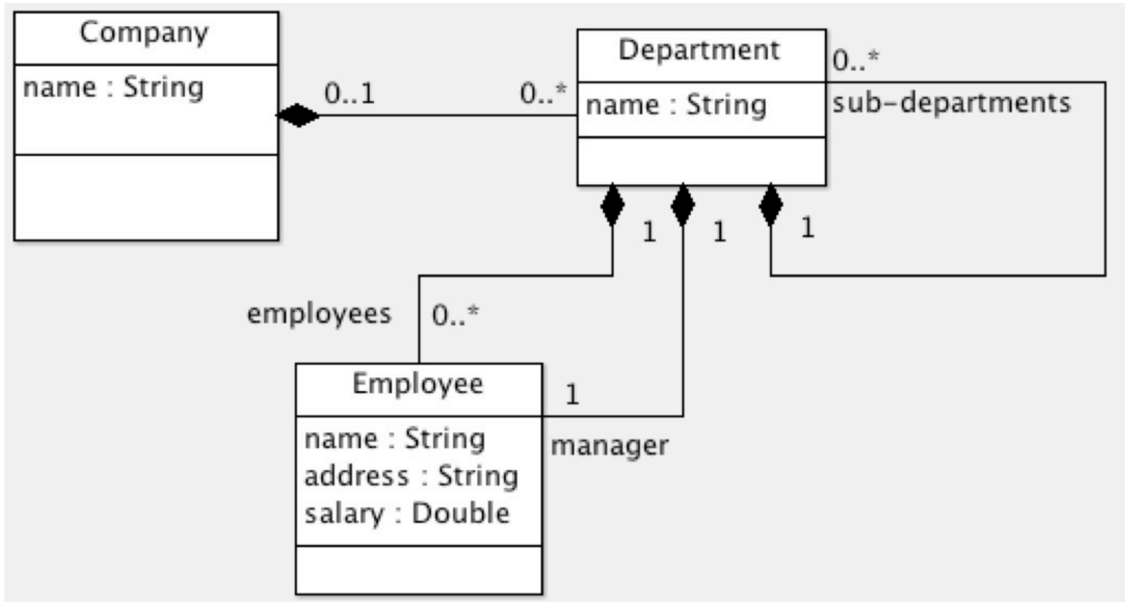
Rosetta Code

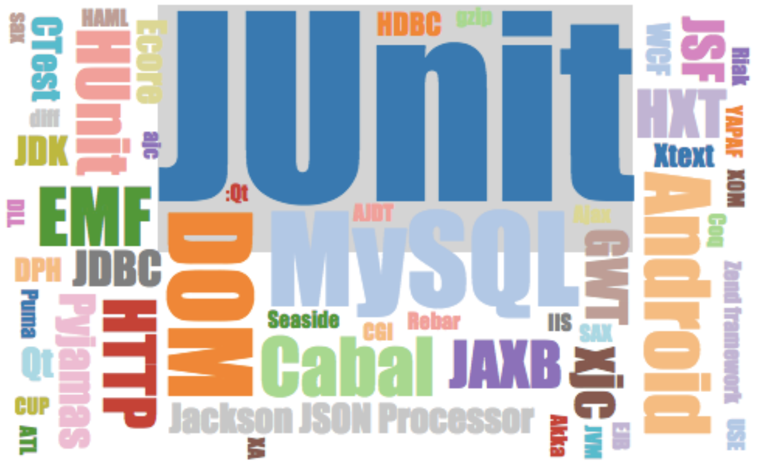
Rosetta Code is a programming chrestomathy site. The idea is to present solutions to the same task in as many different languages as possible, to demonstrate how languages are similar and different, and to aid a person with a grounding in one approach to a problem in learning another. Rosetta Code currently has 850 tasks, 198 draft tasks, and is aware of 651 languages, though we do not (and cannot) have solutions to every task in every language.

Yet another example of a software chrestomathy

https://101wiki.softlang.org/

A HRMS (an information system).
Implemented in diverse languages, technologies, designs.





Characteristics of a software chrestomathy

- Community effort (for aggregation and evaluation)
- Multiplicity of languages
- Infrastructural support
- Revision and access control
- Quality assurance
- Rich metadata
- Process management
- **Reference specification**

How to implement an FSM Language?

A grammar for textual syntax

```
fsm : {state}∗ ;
state : {'initial'}? 'state' stateid '{' {transition}∗ '}' ;
transition : event {'/' action}? {'−>' stateid}? ';' ;
stateid : name ;
event : name ;
action : name ;
```

How to implement an FSM Language?

A metamodel for abstract syntax

```
class fsm { part states : state∗ ; }
class state {
   value initial : boolean ;
   value stateid : string ;
   part transitions : transition∗ ;
}
class transition {
   value event : string ;
   value action : string? ;
   reference target : state ;
}
```

How to implement an FSM Language?

Small-step operational semantics

$$\langle \ldots, \langle b, x, \langle \ldots, \langle e, \langle a \rangle, x' \rangle, \ldots \rangle \rangle, \ldots \rangle \; \vdash \; \langle x, e \rangle \rightarrow \langle x', \langle a \rangle \rangle \qquad [\text{action}]$$

$$\langle \ldots, \langle b, x, \langle \ldots, \langle e, \langle \rangle, x' \rangle, \ldots \rangle \rangle, \ldots \rangle \; \vdash \; \langle x, e \rangle \rightarrow \langle x', \langle \rangle \rangle \qquad [\text{no}-\text{action}]$$

How to implement an FSM Language?

Negative well-formedness test case

```
initial state stateA { eventI/actionI -> stateB; }
state stateB { }
state stateC { }
```

How to implement an FSM Language?

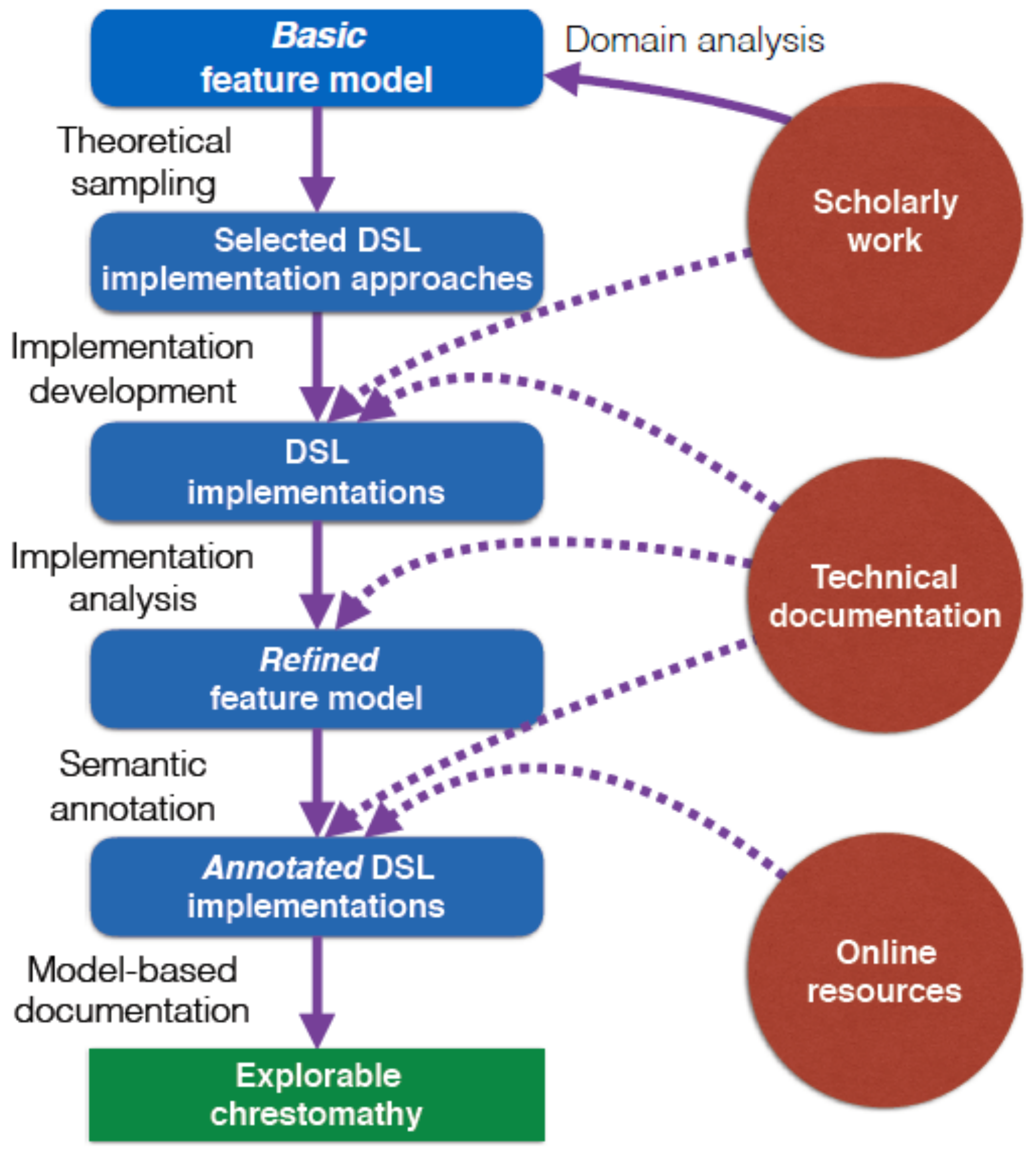
Generated C code

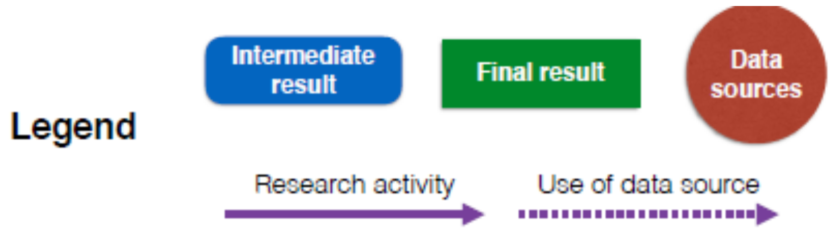
```
enum State {LOCKED,UNLOCKED,EXCEPTION,UNDEFINED};
enum State initial = LOCKED;
enum Event {TICKET,RELEASE,MUTE,PASS};
void alarm() { }
void eject() { }
void collect() { }
enum State next(enum State s, enum Event e) {
    switch(s) {
        case LOCKED:
            switch(e) {
                case TICKET: collect(); return UNLOCKED;
                case PASS: alarm(); return EXCEPTION;
                default: return UNDEFINED;
            }
        case UNLOCKED: ...
        case EXCEPTION: ...
        default: return UNDEFINED;
    }
}
```

‘How’

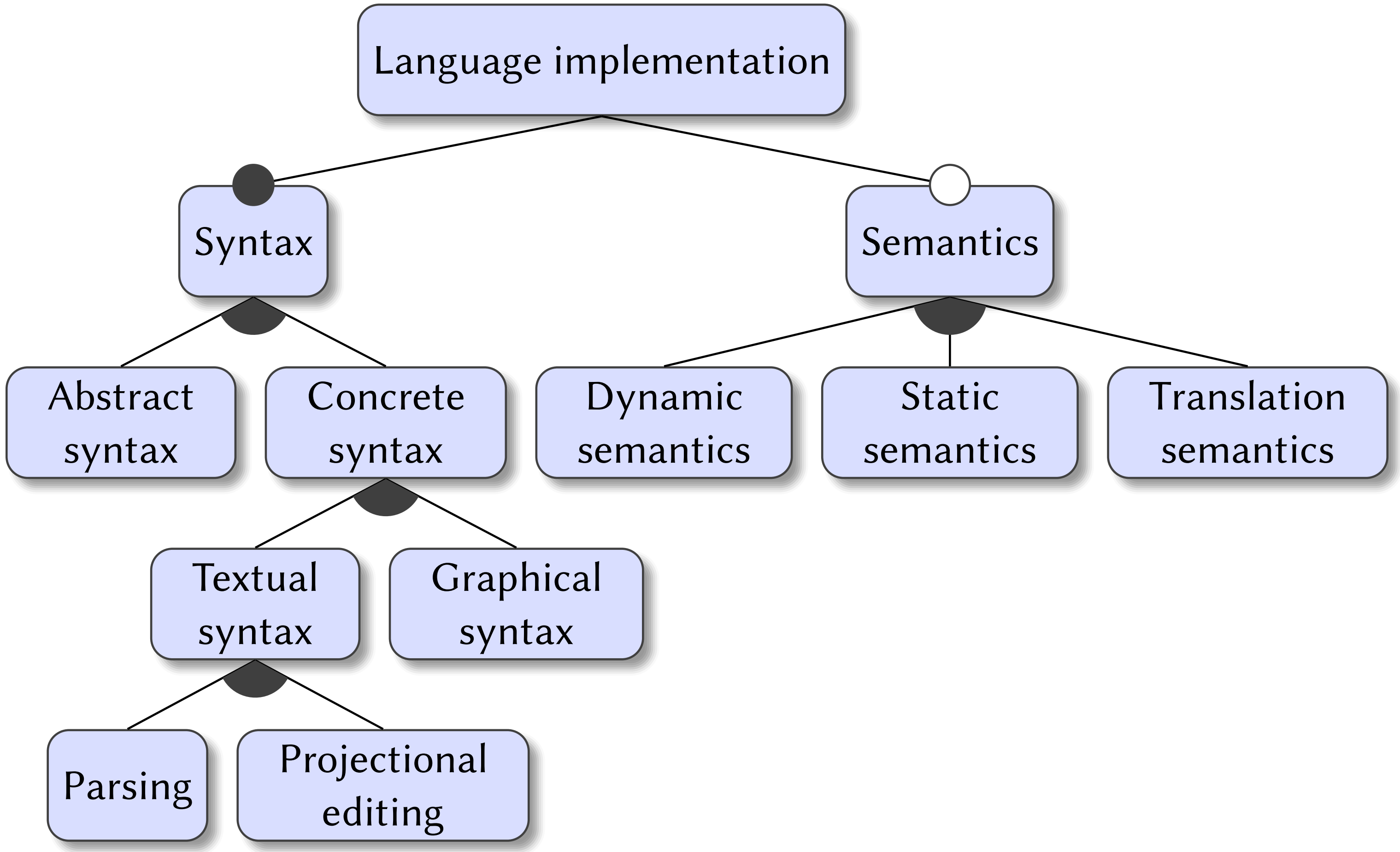
What is the MetaLib methodology?

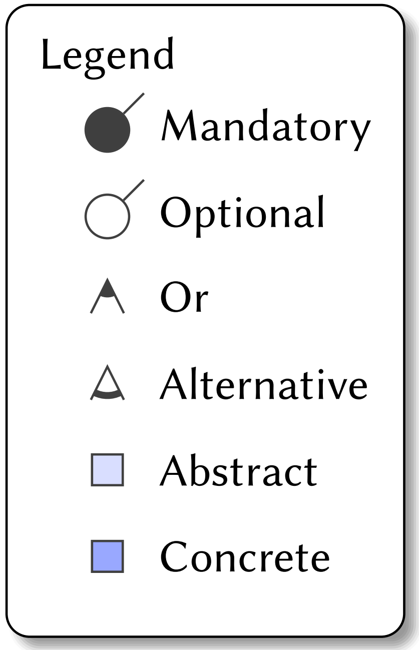
The collected implementations are organized and documented with the help of feature modeling, semantic annotations, and model-based documentation.





Domain analysis





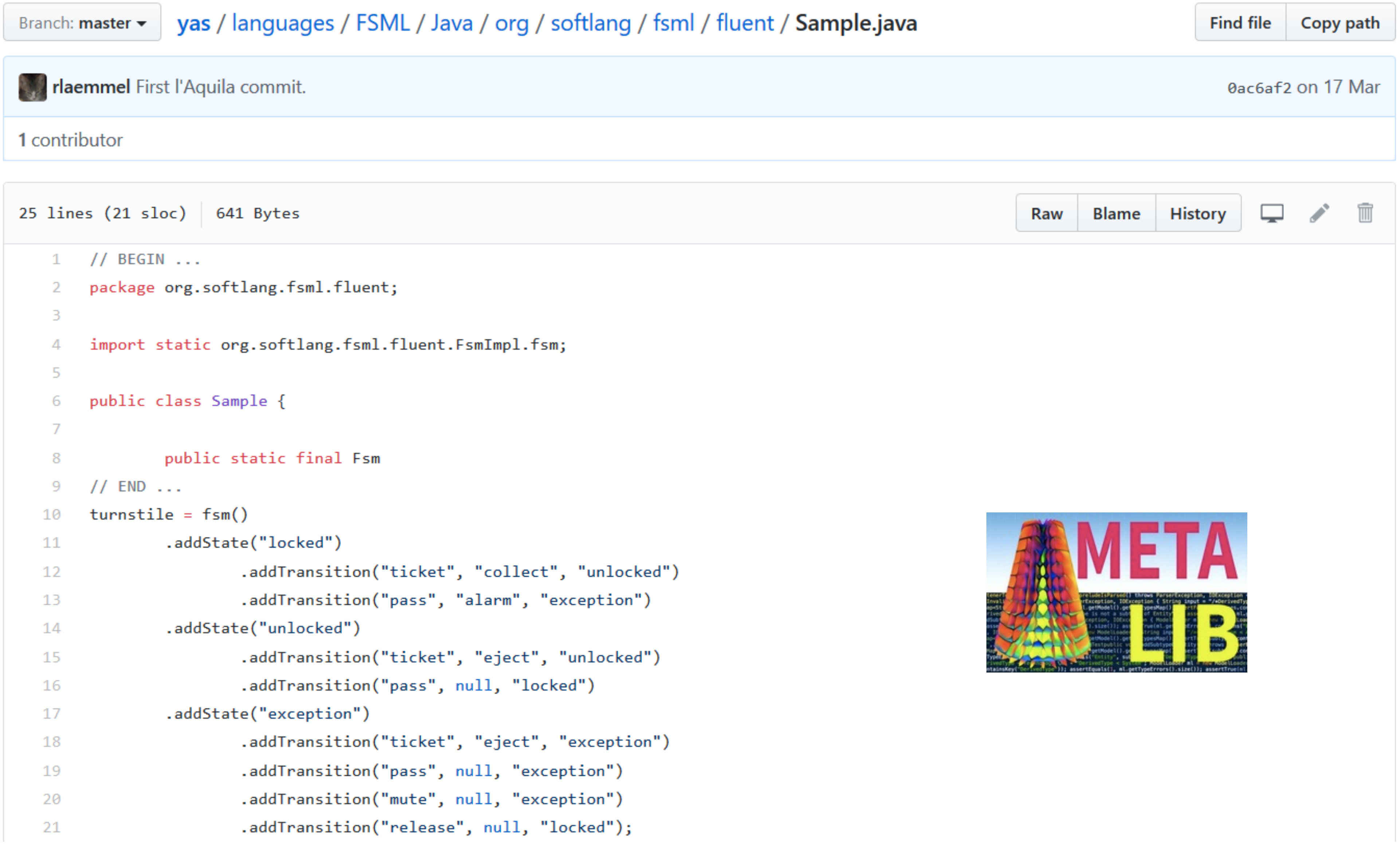
Theoretical sampling

Chrestomathy member	Languages & technologies
javaInfluentInternal	Java
javaFluentInternal	Java
javaExternal	Java, ANTLR
pythonInternal	Python, Graphviz
pythonExternal	Python, ANTLR
haskellQuasiQuotation	Haskell (+TH+QQ)
scalaEmbedded	Scala
mps	MPS
spoofax	Spoofax
racket	Racket
rascal	Rascal
emfXMI	EMF
emfSirius	EMF, Sirius
emfXtext	EMF, Xtext

Coverage of

- **mainstream languages;**
- **programming paradigms;**
- **DSL implementation styles;**
- **technological spaces;**
- **the basic feature model.**

Implementation development



Branch: master yas / languages / FSML / Java / org / softlang / fsml / fluent / Sample.java Find file Copy path

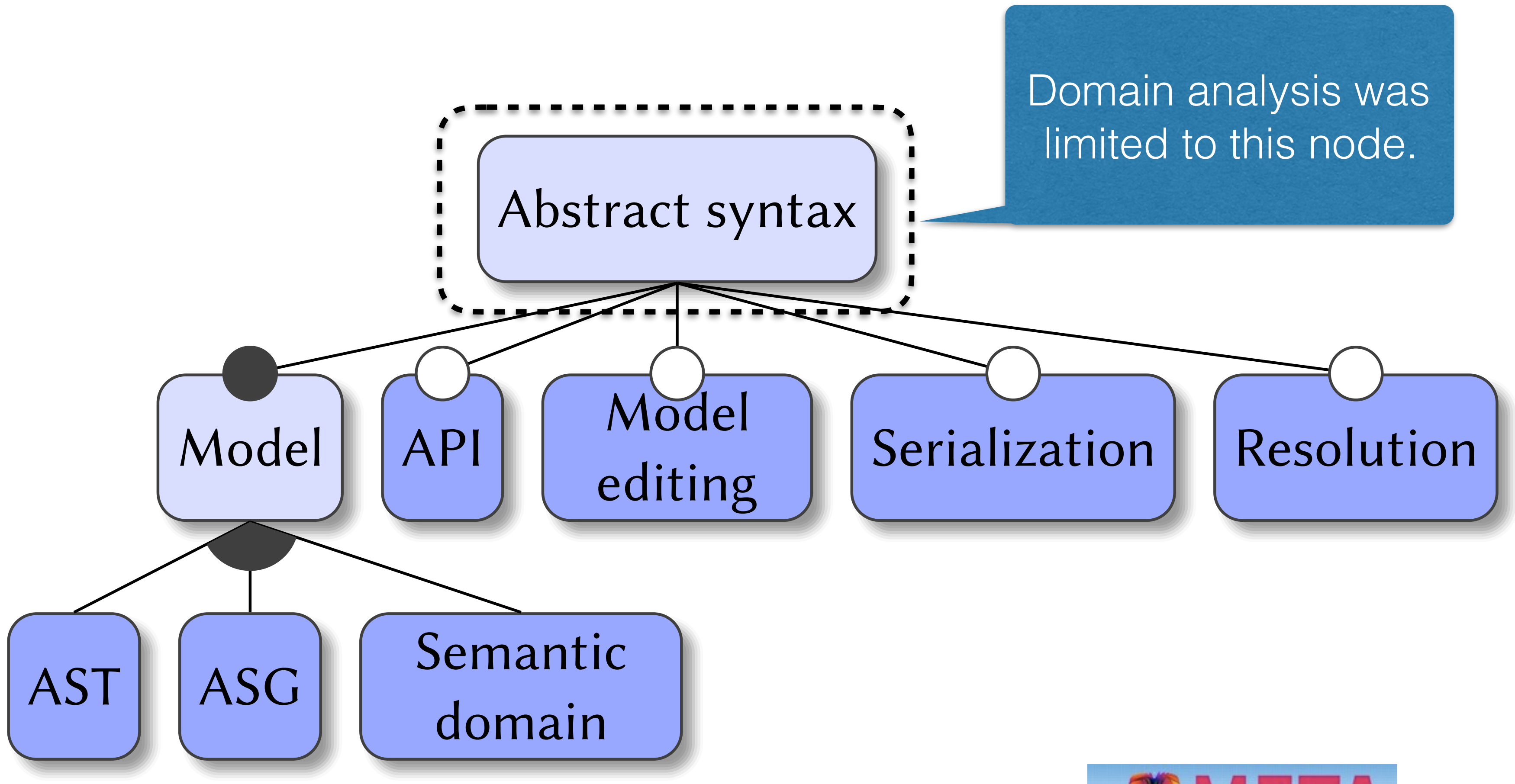
rlaemmel First l'Aquila commit. 0ac6af2 on 17 Mar

1 contributor

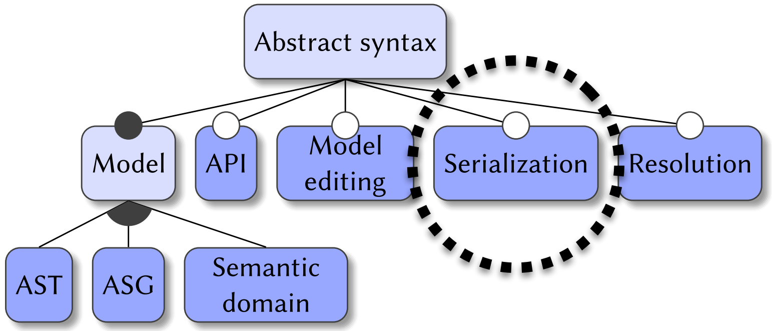
25 lines (21 sloc) | 641 Bytes Raw Blame History

```
// BEGIN ...
package org.softlang.fsml.fluent;

import static org.softlang.fsml.fluent.FsmImpl.fsm;

public class Sample {

        public static final Fsm
// END ...
turnstile = fsm()
        .addState("locked")
                .addTransition("ticket", "collect", "unlocked")
                .addTransition("pass", "alarm", "exception")
        .addState("unlocked")
                .addTransition("ticket", "eject", "unlocked")
                .addTransition("pass", null, "locked")
        .addState("exception")
                .addTransition("ticket", "eject", "exception")
                .addTransition("pass", null, "exception")
                .addTransition("mute", null, "exception")
                .addTransition("release", null, "locked");
```

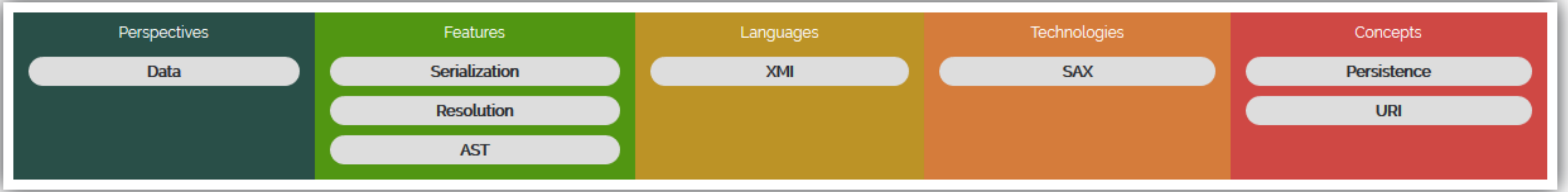
Implementation analysis



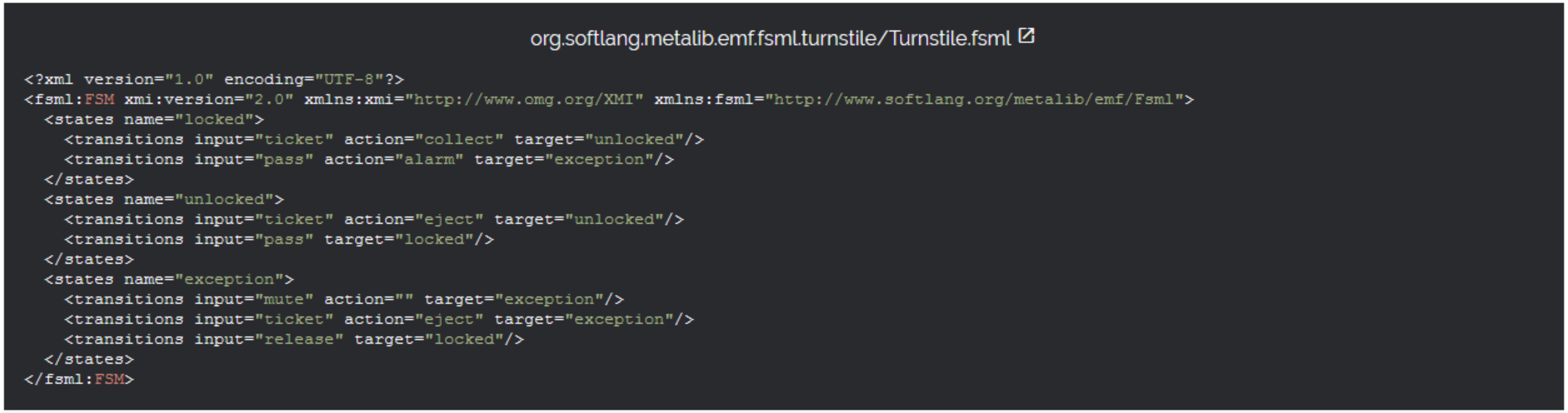
Implementation analysis



The turnstile object in EMF's textual exchange format



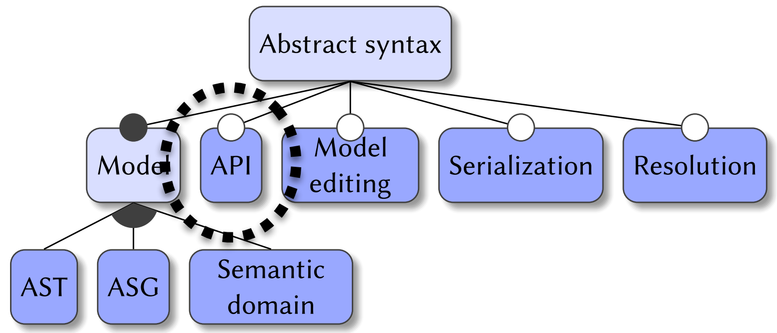
Perspectives	Features	Languages	Technologies	Concepts
Data	Serialization	XMI	SAX	Persistence
	Resolution			URI
	AST			



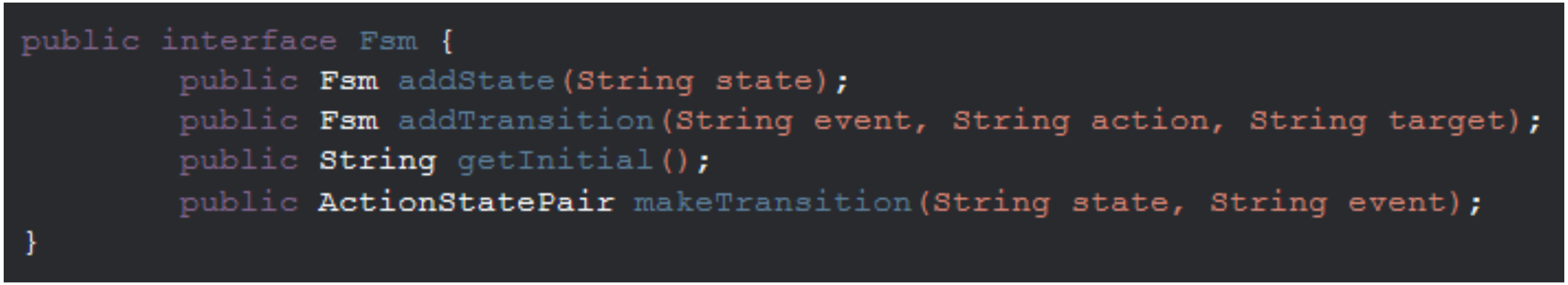
org.softlang.metalib.emf.fsml.turnstile/Turnstile.fsml

```
<?xml version="1.0" encoding="UTF-8"?>
<fsml:FSM xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:fsml="http://www.softlang.org/metalib/emf/Fsml">
  <states name="locked">
    <transitions input="ticket" action="collect" target="unlocked"/>
    <transitions input="pass" action="alarm" target="exception"/>
  </states>
  <states name="unlocked">
    <transitions input="ticket" action="eject" target="unlocked"/>
    <transitions input="pass" target="locked"/>
  </states>
  <states name="exception">
    <transitions input="mute" action="" target="exception"/>
    <transitions input="ticket" action="eject" target="exception"/>
    <transitions input="release" target="locked"/>
  </states>
</fsml:FSM>
```

Implementation analysis

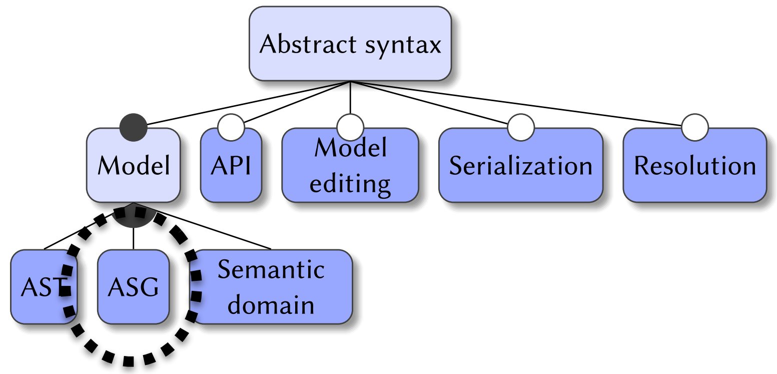


…/fluent/…/Fsm.java

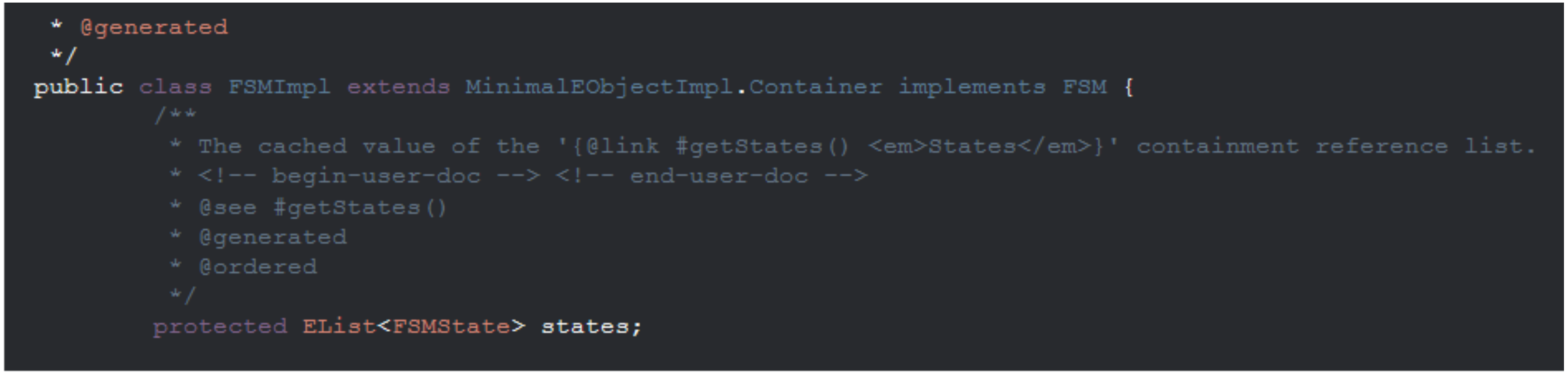


```
public interface Fsm {
        public Fsm addState(String state);
        public Fsm addTransition(String event, String action, String target);
        public String getInitial();
        public ActionStatePair makeTransition(String state, String event);
}
```

Implementation analysis

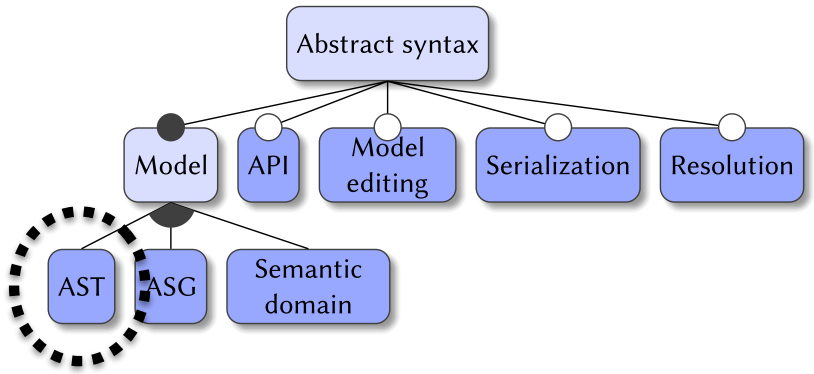


…/emf/…/FSMImpl.java

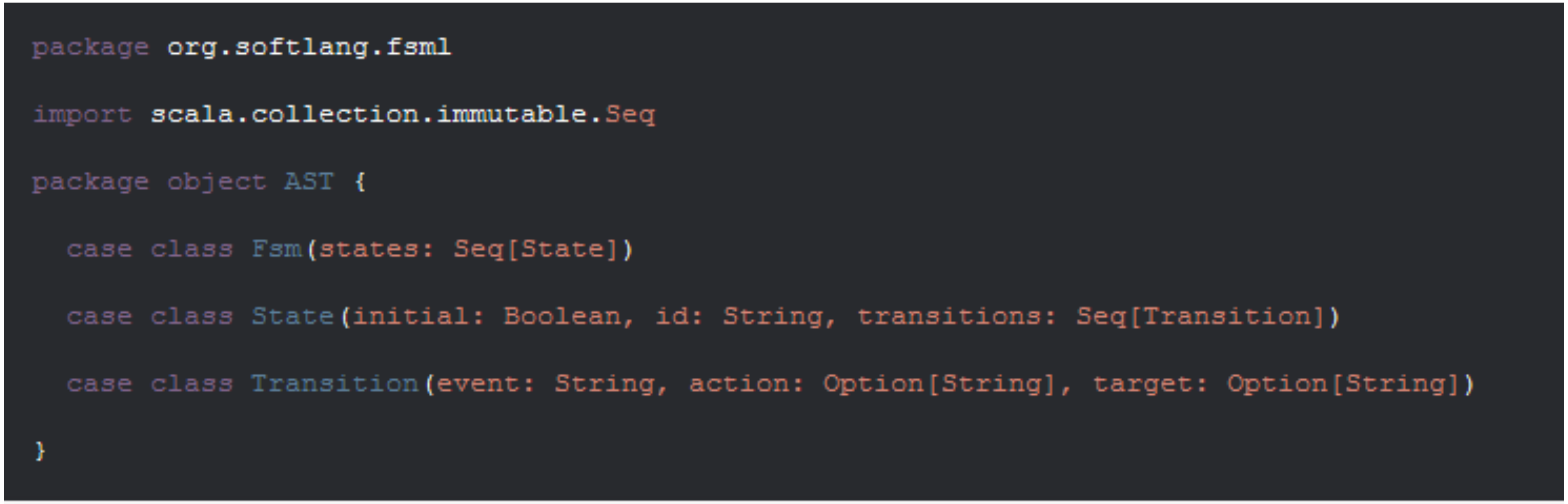


```
 * @generated
 */
public class FSMImpl extends MinimalEObjectImpl.Container implements FSM {
	/**
	 * The cached value of the '{@link #getStates() <em>States</em>}' containment reference list.
	 * <!-- begin-user-doc --> <!-- end-user-doc -->
	 * @see #getStates()
	 * @generated
	 * @ordered
	 */
	protected EList<FSMState> states;
```

Implementation analysis

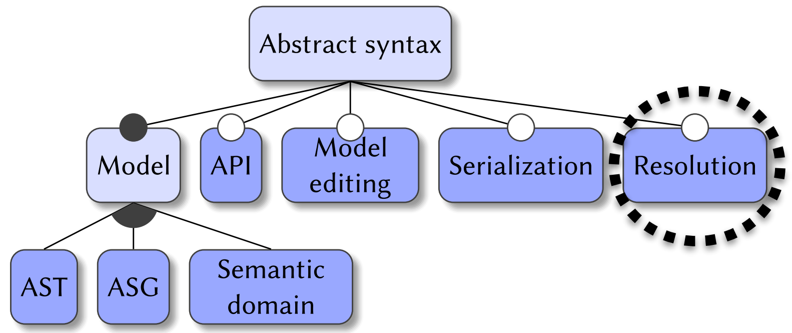


…/AST.scala

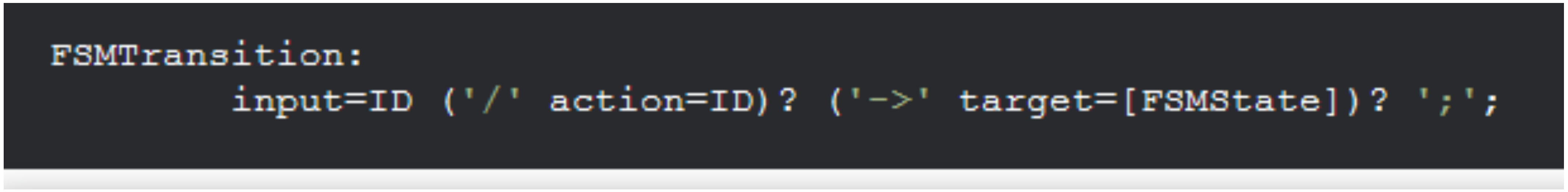


```
package org.softlang.fsml

import scala.collection.immutable.Seq

package object AST {

  case class Fsm(states: Seq[State])

  case class State(initial: Boolean, id: String, transitions: Seq[Transition])

  case class Transition(event: String, action: Option[String], target: Option[String])

}
```

Implementation analysis

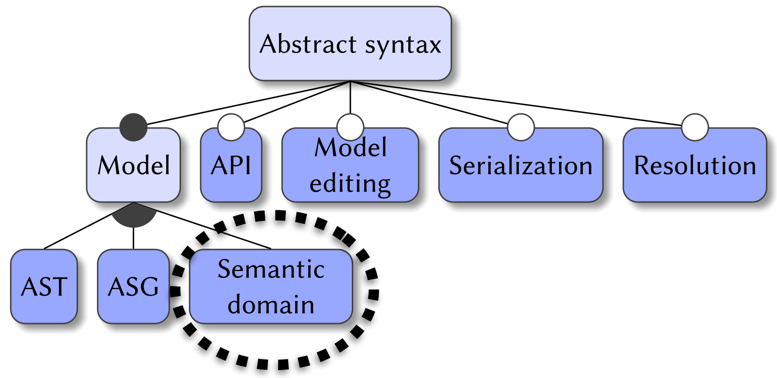


…/FSML.xtext

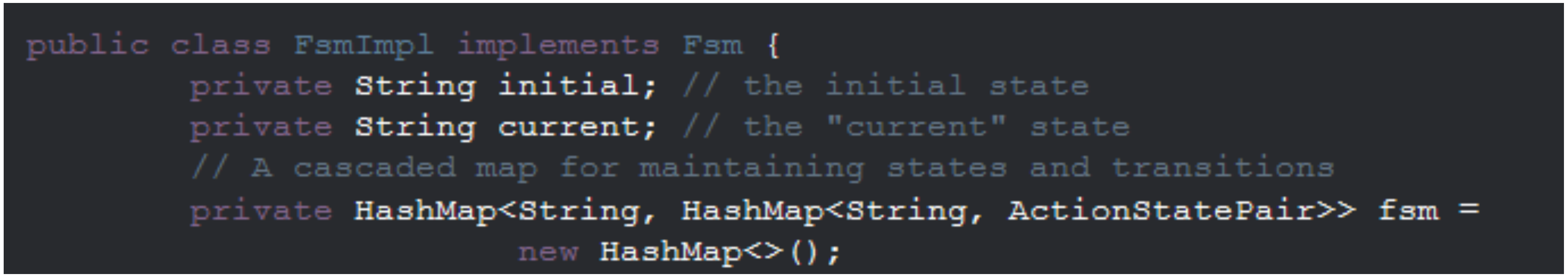


```
FSMTransition:
        input=ID ('/' action=ID)? ('->' target=[FSMState])? ';';
```

Implementation analysis



…/fluent/…/FsmImpl.java



```
public class FsmImpl implements Fsm {
        private String initial; // the initial state
        private String current; // the "current" state
        // A cascaded map for maintaining states and transitions
        private HashMap<String, HashMap<String, ActionStatePair>> fsm =
                new HashMap<>();
```

Implementation analysis

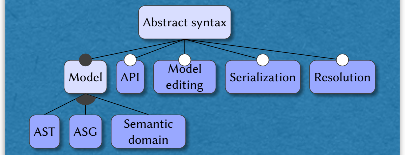
	emfSirius	emfXMI	emfXtext	haskellQuasiQuotation	javaExternal	javaFluentInternal	javaInfluentInternal	mps	pythonExternal	pythonInternal	racket	rascal	scalaEmbedded	spoofax
Abstract syntax	×	×	×	×	×	×	×	×	×	×	×	×	×	
AST	×	×	×	×	×		×	×	×	×	×	×	×	
ASG	×	×	×											
Semantic domain						×								
Model editing	×	×												
API	×	×	×			×	×		×	×		×		
Serialization	×	×							×					
Resolution	×	×	×											
Textual syntax			×	×	×			×	×		×	×	×	×
Text-to-CST					×							×		×
Text-to-AST			×	×	×				×				×	
Text-to-ASG			×											
Projectional editing								×						
Scanning			×	×	×				×					
Abstraction					×									

Implementation analysis

	emfSirius	emfXMI	emfXtext	haskellQuasiQuotation	javaExternal	javaFluentInternal	javaInfluentInternal	mps	pythonExternal	pythonInternal	racket	rascal	scalaEmbedded	spoofax
Abstract syntax	×	×	×	×	×	×	×	×	×	×	×	×	×	
AST	×	×	×	×	×		×	×	×	×	×	×	×	
ASG	×	×	×											
Semantic domain						×								
Model editing	×	×												
API	×	×	×			×	×		×	×		×		
Serialization	×	×							×					
Resolution	×	×	×											
Textual syntax			×	×	×			×	×		×	×	×	×
Text-to-CST					×							×		×
Text-to-AST			×	×	×				×				×	
Text-to-ASG			×											
Projectional editing								×						
Scanning			×	×	×				×					
Abstraction					×									
Replacement								×			×		×	
Graphical syntax	×	×	×							×		×		
Graph rendering	×		×							×		×		
Graph editing	×	×												
Dynamic semantics						×	×			×	×			
Interpretation						×	×			×	×			
Static semantics		×	×	×				×		×	×	×	×	×
Analysis		×	×	×				×		×	×	×	×	×
Piggyback				×							×		×	
Translation semantics			×	×			×	×		×		×	×	×
Compilation			×				×	×		×		×		×
Staging				×									×	

Semantic annotation

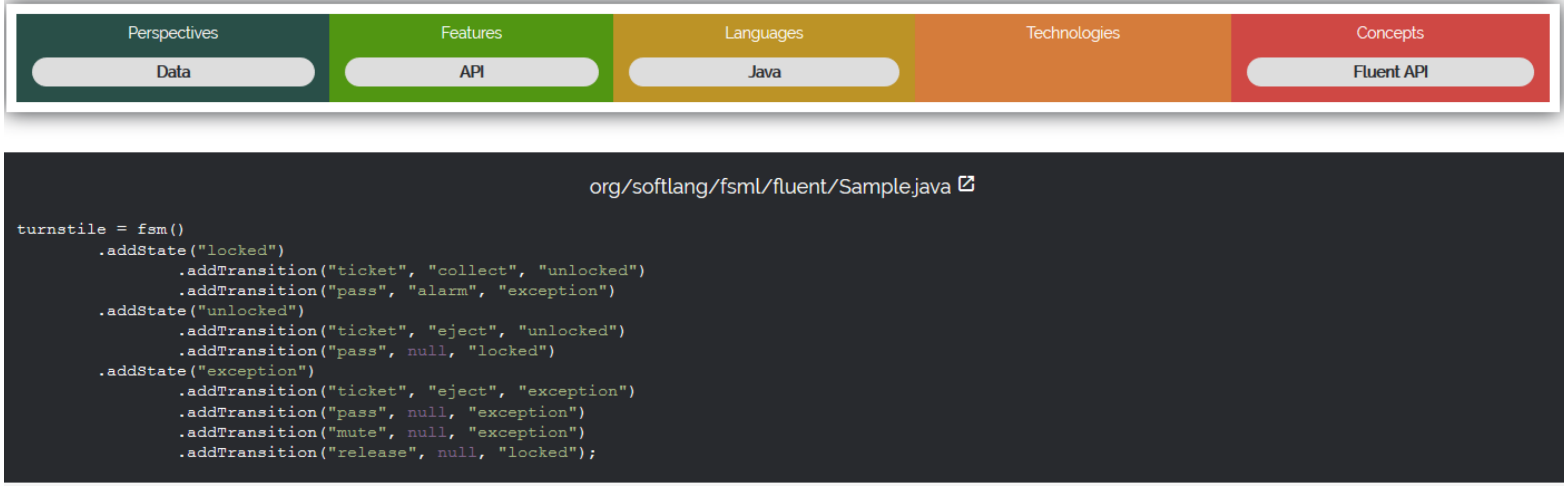
Data
Implementation
Test
Capture



Python
Java
XML
C
...

JUnit
ANTLR3
Acceleo
...

Fluent API
Command
Macro
...

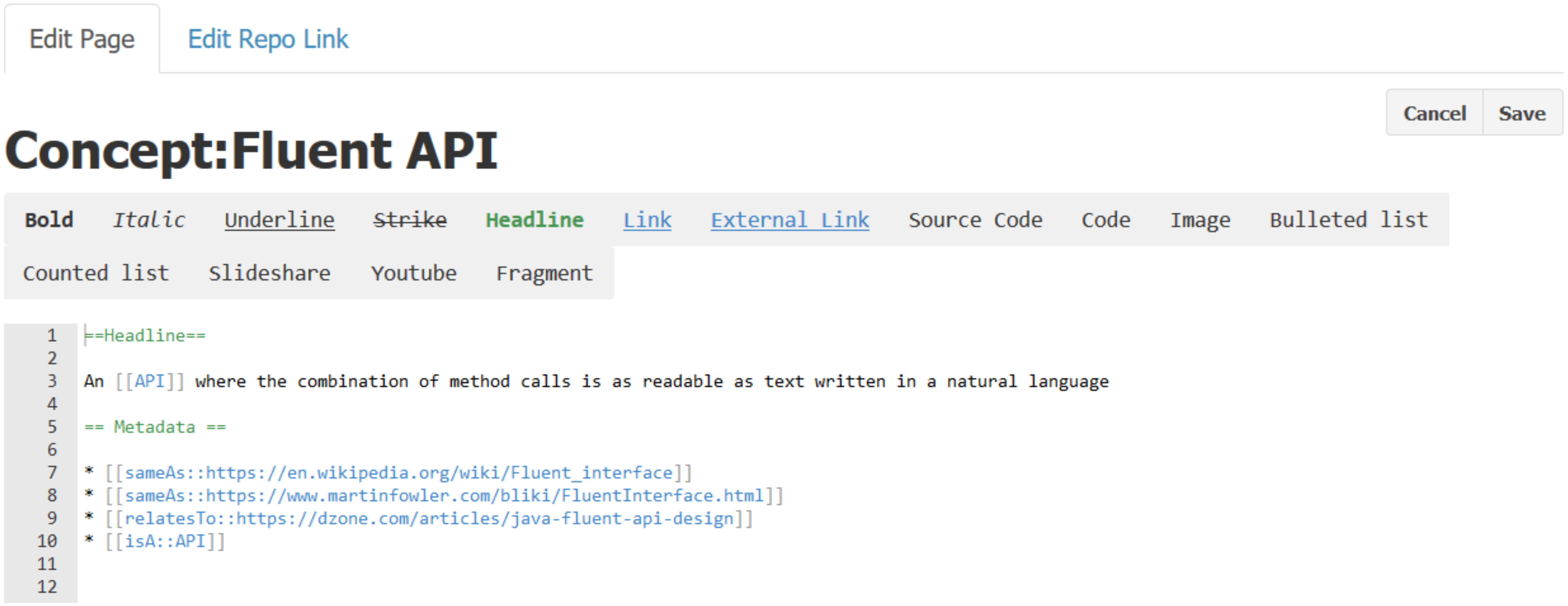


Perspectives	Features	Languages	Technologies	Concepts
Data	API	Java		Fluent API

org/softlang/fsml/fluent/Sample.java

```
turnstile = fsm()
        .addState("locked")
                .addTransition("ticket", "collect", "unlocked")
                .addTransition("pass", "alarm", "exception")
        .addState("unlocked")
                .addTransition("ticket", "eject", "unlocked")
                .addTransition("pass", null, "locked")
        .addState("exception")
                .addTransition("ticket", "eject", "exception")
                .addTransition("pass", null, "exception")
                .addTransition("mute", null, "exception")
                .addTransition("release", null, "locked");
```

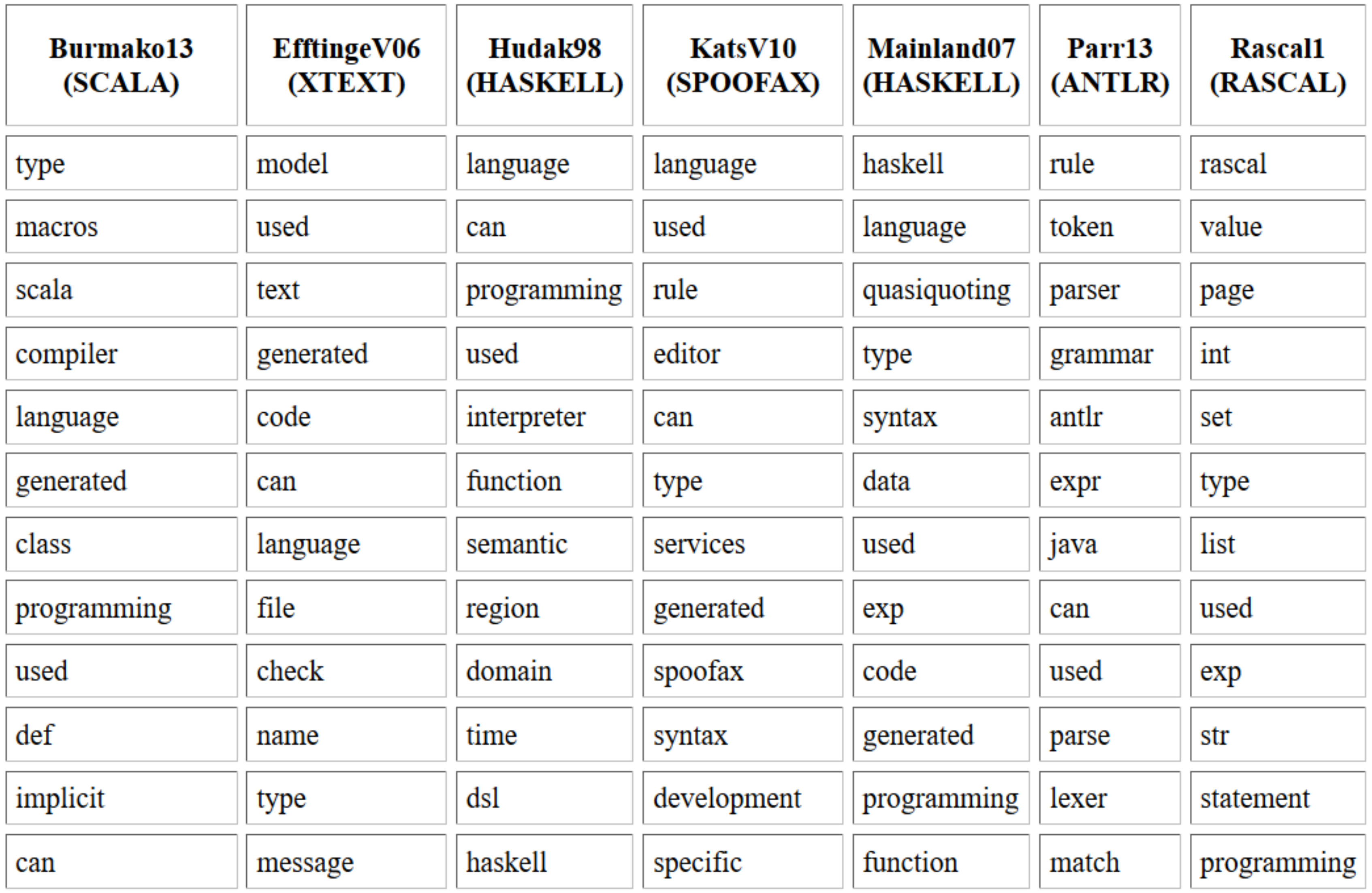
Semantic annotation



Concept:Fluent API

```
==Headline==

An [[API]] where the combination of method calls is as readable as text written in a natural language

== Metadata ==

* [[sameAs::https://en.wikipedia.org/wiki/Fluent_interface]]
* [[sameAs::https://www.martinfowler.com/bliki/FluentInterface.html]]
* [[relatesTo::https://dzone.com/articles/java-fluent-api-design]]
* [[isA::API]]
```

Information retrieval informing implementation analysis and semantic annotation



Burmako13 (SCALA)	EfftingeV06 (XTEXT)	Hudak98 (HASKELL)	KatsV10 (SPOOFAX)	Mainland07 (HASKELL)	Parr13 (ANTLR)	Rascal1 (RASCAL)
type	model	language	language	haskell	rule	rascal
macros	used	can	used	language	token	value
scala	text	programming	rule	quasiquoting	parser	page
compiler	generated	used	editor	type	grammar	int
language	code	interpreter	can	syntax	antlr	set
generated	can	function	type	data	expr	type
class	language	semantic	services	used	java	list
programming	file	region	generated	exp	can	used
used	check	domain	spoofax	code	used	exp
def	name	time	syntax	generated	parse	str
implicit	type	dsl	development	programming	lexer	statement
can	message	haskell	specific	function	match	programming

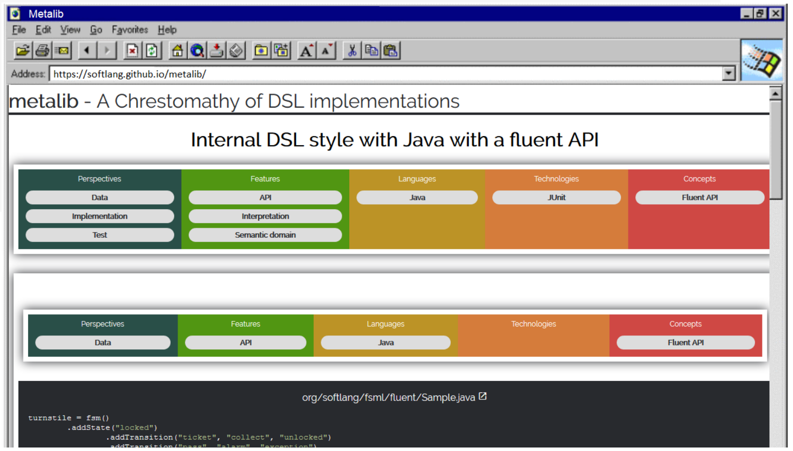
Semantic annotation

Boilerplate code
Fluent API
Lexer
Interpolation
Inline Macro
Name binding
Exception
Internal DSL
OO interface
Listener
Projectional Editing
Dictionary
Static annotation
Case Class
Viewpoint
Factory
Transformation Rule
Visitor
Tree Walker
Adapter
Parser combinator
Lifting
External DSL
Reduction
Struct
Acceptor
Token
Parser
Palette
Persistence
IDE
OO class
Code generation
Structured Editor
Batch file
Command
Semantic action
Language Extension
Algebraic data type
Grammar
Package
Metamodel
Implicit conversion
URI
Reference
Parser generation
Record type
Macro
Quote
Tail recursion

Model-based documentation

https://github.com/softlang/metalib/blob/master/models/javaInfluentInternal.json

```
{ "name": "javaFluentInternal",
  "baseuri": "https://github.com/softlang/yas/tree/master/languages/
          FSML/Java/org/softlang",
  "headline": "Internal DSL style with Java with a fluent API",
  "sections": [
    { "features": ["API" ],
      "perspectives": ["data"],
      "languages": ["Java"],
      "concepts": ["Fluent API"],
      "technologies": [],
      "artifacts": [{ "type": "all", "link": "fsml/fluent/Sample.java"}]
    },
    ...
  ]
}
```



Metalib

Address: https://softlang.github.io/metalib/

metalib - A Chrestomathy of DSL implementations

Internal DSL style with Java with a fluent API

Perspectives	Features	Languages	Technologies	Concepts
Data	API	Java	JUnit	Fluent API
Implementation	Interpretation			
Test	Semantic domain			

Perspectives	Features	Languages	Technologies	Concepts
Data	API	Java		Fluent API

org/softlang/fsml/fluent/Sample.java

```
turnstile = fsm()
        .addState("locked")
                .addTransition("ticket", "collect", "unlocked")
```

Metamodel of documentation

```
// Documentation of contributions
class document {
   value name : string; // The name of the contribution
   value headline : string; // A one−liner explanation
   value baseuri : string; // Base URI for links
   part sections : section+; // Sections of the documentation
}

// Sections in a documentation
class section {
   value headline : string?; // Optional one−liner explanation
   part perspectives : perspective+; // Perspective of section
   value features : string+; // Features addressed by section
   value languages : string*; // Languages used
   value technologies : string*; // Technologies used
   value concepts : string*; // Concepts used
   part artifacts : artifact+; // Artifacts to be shown
}
```

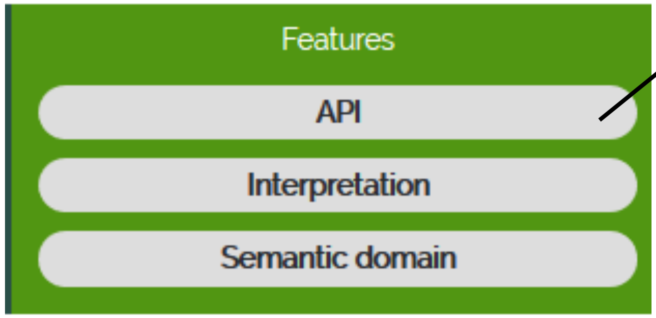
```
}

// Perspectives of documentation
enum perspective {
   implementation, // i.e., feature implementation
   data, // e.g., instance of grammar or metamodel
   test, // i.e., application of implementation
   build, // e.g., code generator application
   capture // e.g., screenshot or session log
}

// Artifacts for projected by section
abstract class artifact {
   value link : string; // A relative URI
   value format : string; // MIME−like format type
}
class none extends artifact { } // Nothing to show
class all extends artifact { } // All to show
class some extends artifact { // A specific line range to show
   value from : integer;
   value to : integer;
}
```

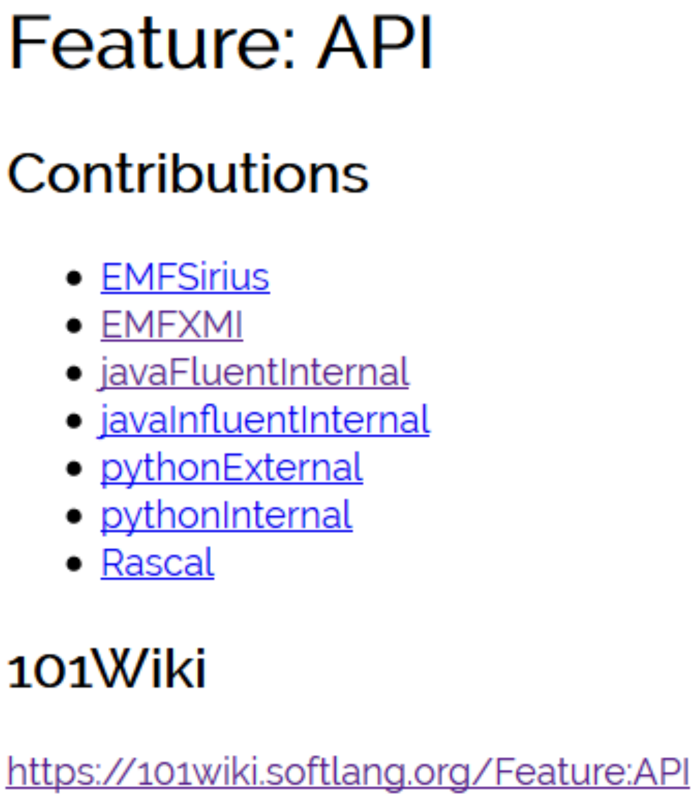
‘Why’

What scenarios for learning and teaching exist?

The chrestomathy is useful for learning (teaching) in so far that it provides a high level of abstraction for metaprogramming and it directly enables the side-by-side exploration of implementation approaches for DSLs (so that one can learn new metaprogramming techniques based on techniques already known).

Which DSL implementation uses an API?




Feature: API

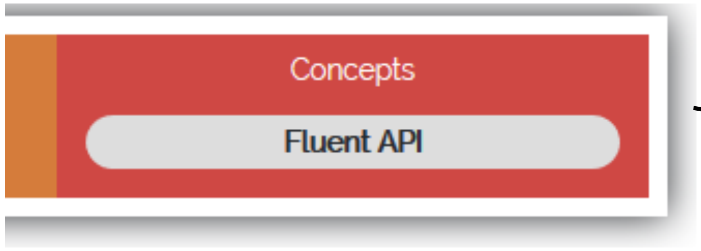
Contributions

- EMFSirius
- EMFXMI
- javaFluentInternal
- javaInfluentInternal
- pythonExternal
- pythonInternal
- Rascal

101Wiki

https://101wiki.softlang.org/Feature:API

What is a fluent API?



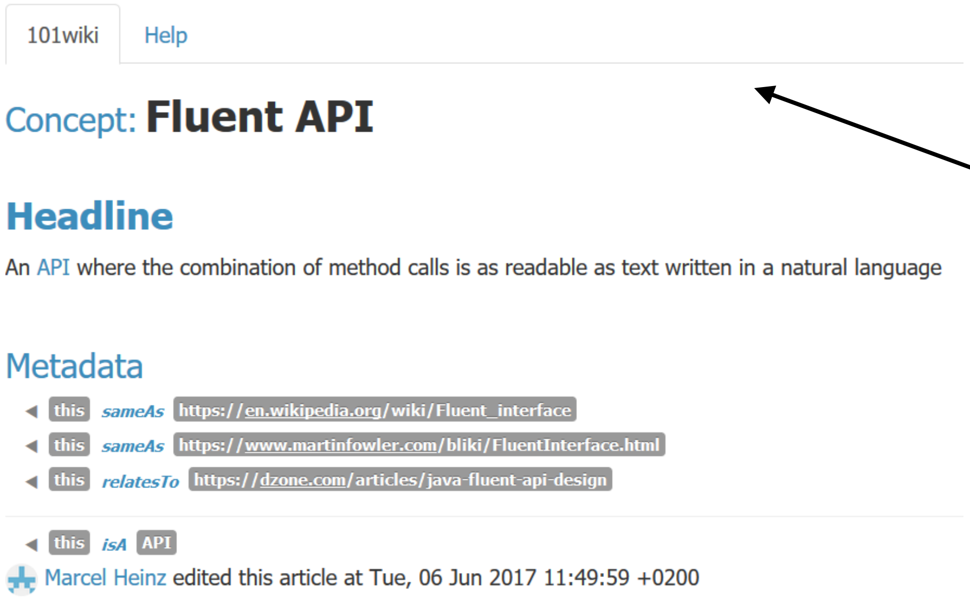
Concept:Fluent API

Contributions

- javaFluentInternal
- pythonInternal
- Rascal

101Wiki

https://101wiki.softlang.org/Fluent API



101wiki Help

Concept: **Fluent API**

Headline

An API where the combination of method calls is as readable as text written in a natural language

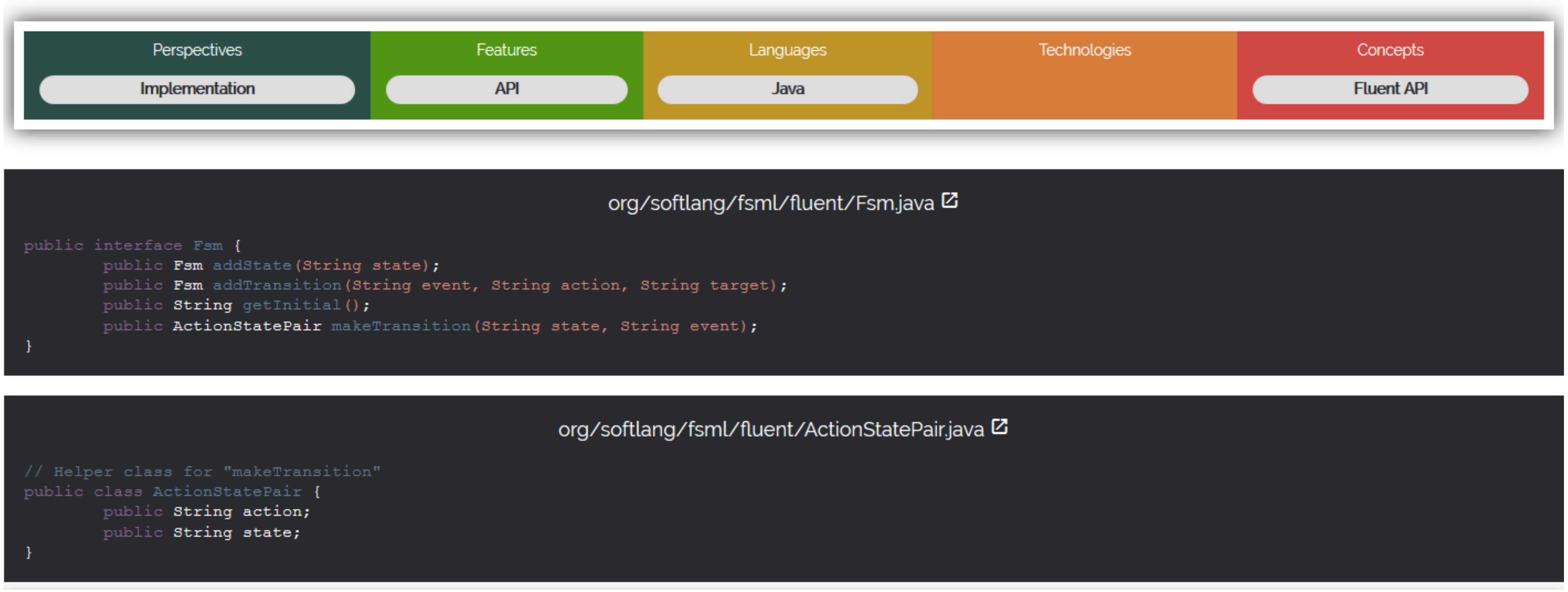
Metadata

- this *sameAs* https://en.wikipedia.org/wiki/Fluent_interface
- this *sameAs* https://www.martinfowler.com/bliki/FluentInterface.html
- this *relatesTo* https://dzone.com/articles/java-fluent-api-design

- this *isA* API

Marcel Heinz edited this article at Tue, 06 Jun 2017 11:49:59 +0200

Where is the API implemented?



org/softlang/fsml/fluent/Fsm.java

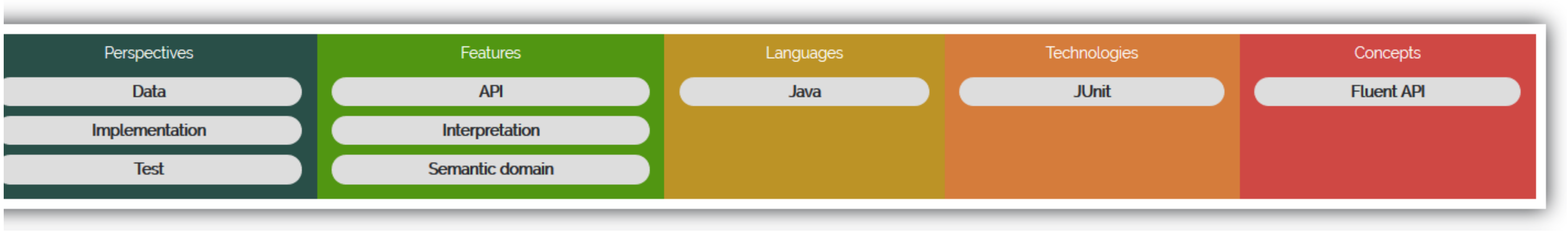
```
public interface Fsm {
        public Fsm addState(String state);
        public Fsm addTransition(String event, String action, String target);
        public String getInitial();
        public ActionStatePair makeTransition(String state, String event);
}
```

org/softlang/fsml/fluent/ActionStatePair.java

```
// Helper class for "makeTransition"
public class ActionStatePair {
        public String action;
        public String state;
}
```

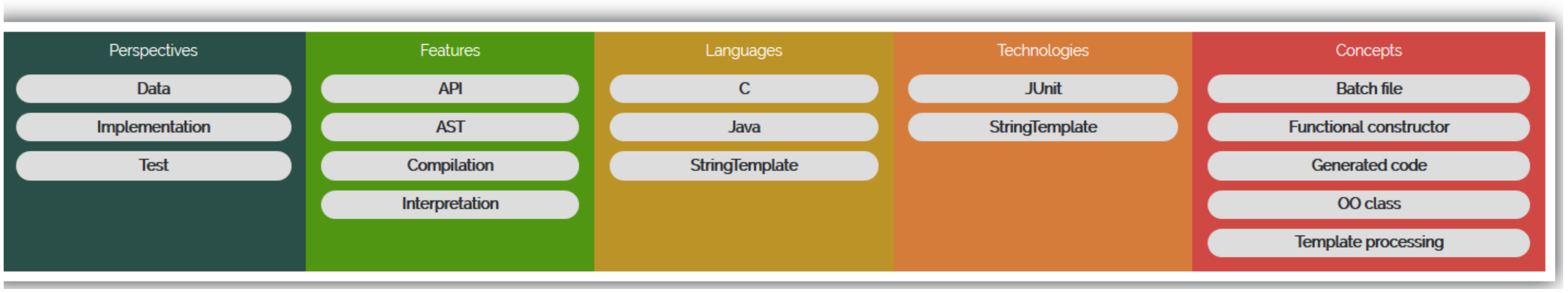
How does influent differ from fluent java implementation?

Internal DSL style with Java with a fluent API



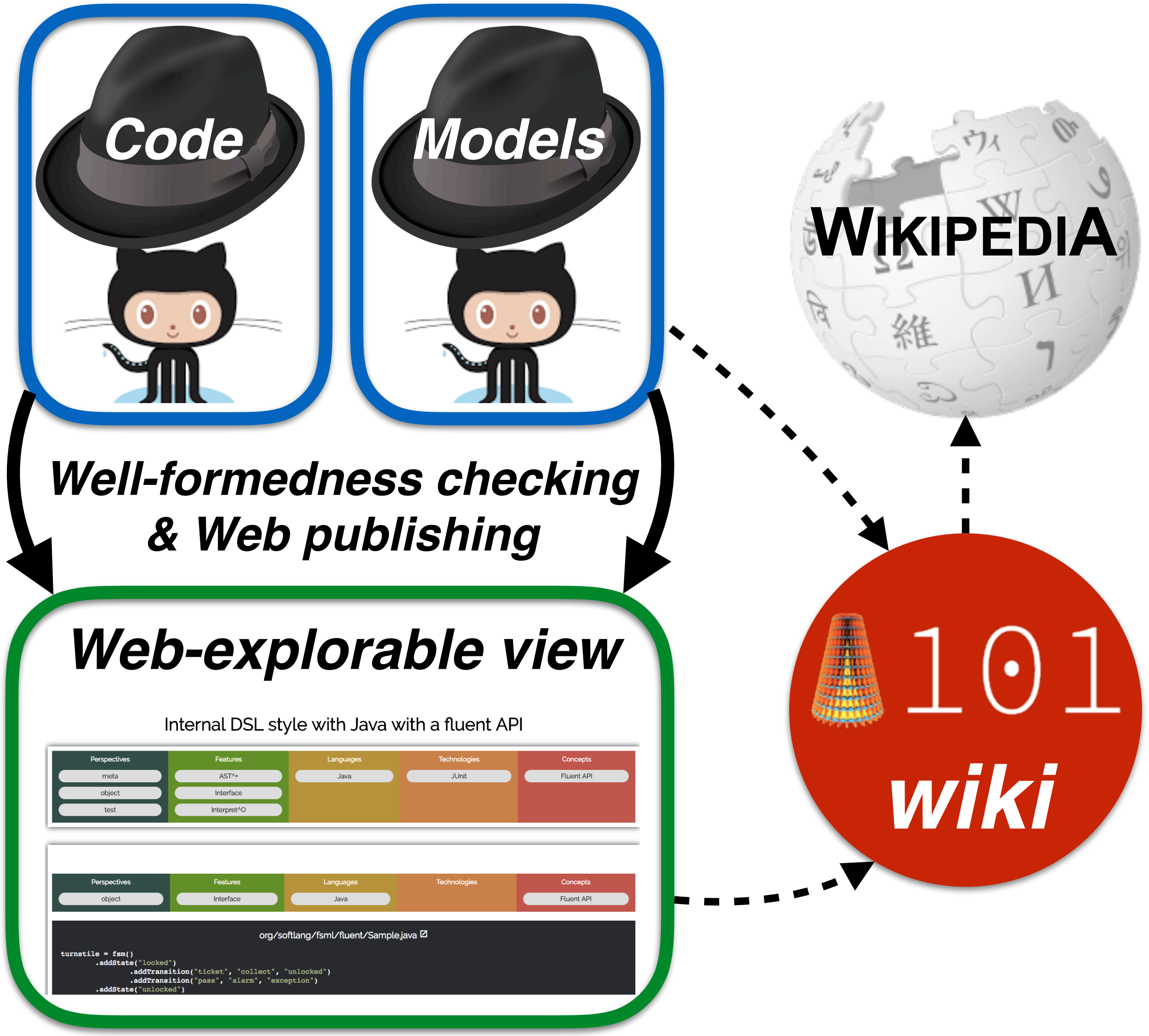
Perspectives	Features	Languages	Technologies	Concepts
Data	API	Java	JUnit	Fluent API
Implementation	Interpretation			
Test	Semantic domain			

Internal DSL style with Java and an influent API



Perspectives	Features	Languages	Technologies	Concepts
Data	API	C	JUnit	Batch file
Implementation	AST	Java	StringTemplate	Functional constructor
Test	Compilation	StringTemplate		Generated code
	Interpretation			OO class
				Template processing

The documentation approach



Internal DSL style with Java with a fluent API

Perspectives	Features	Languages	Technologies	Concepts
meta	AST^+	Java	JUnit	Fluent API
object	Interface			
test	Interpret^O			

Perspectives	Features	Languages	Technologies	Concepts
object	Interface	Java		Fluent API

org/softlang/fsml/fluent/Sample.java

```
turnstile = fsm()
        .addState("locked")
                .addTransition("ticket", "collect", "unlocked")
                .addTransition("pass", "alarm", "exception")
        .addState("unlocked")
```

Future work

- Add **contributions**.
- Add **features**.
- Refine **theoretical sampling**.
- Advance the use of **IR techniques**.
- Define and improve quality of **101wiki**.
- **Cross-validate** contributions and documentation.
- Evaluate MetaLib in **classroom**.

A Chrestomathy of DSL Implementations

Thanks!
Questions?
Comments?

http://www.softlang.org/metalib