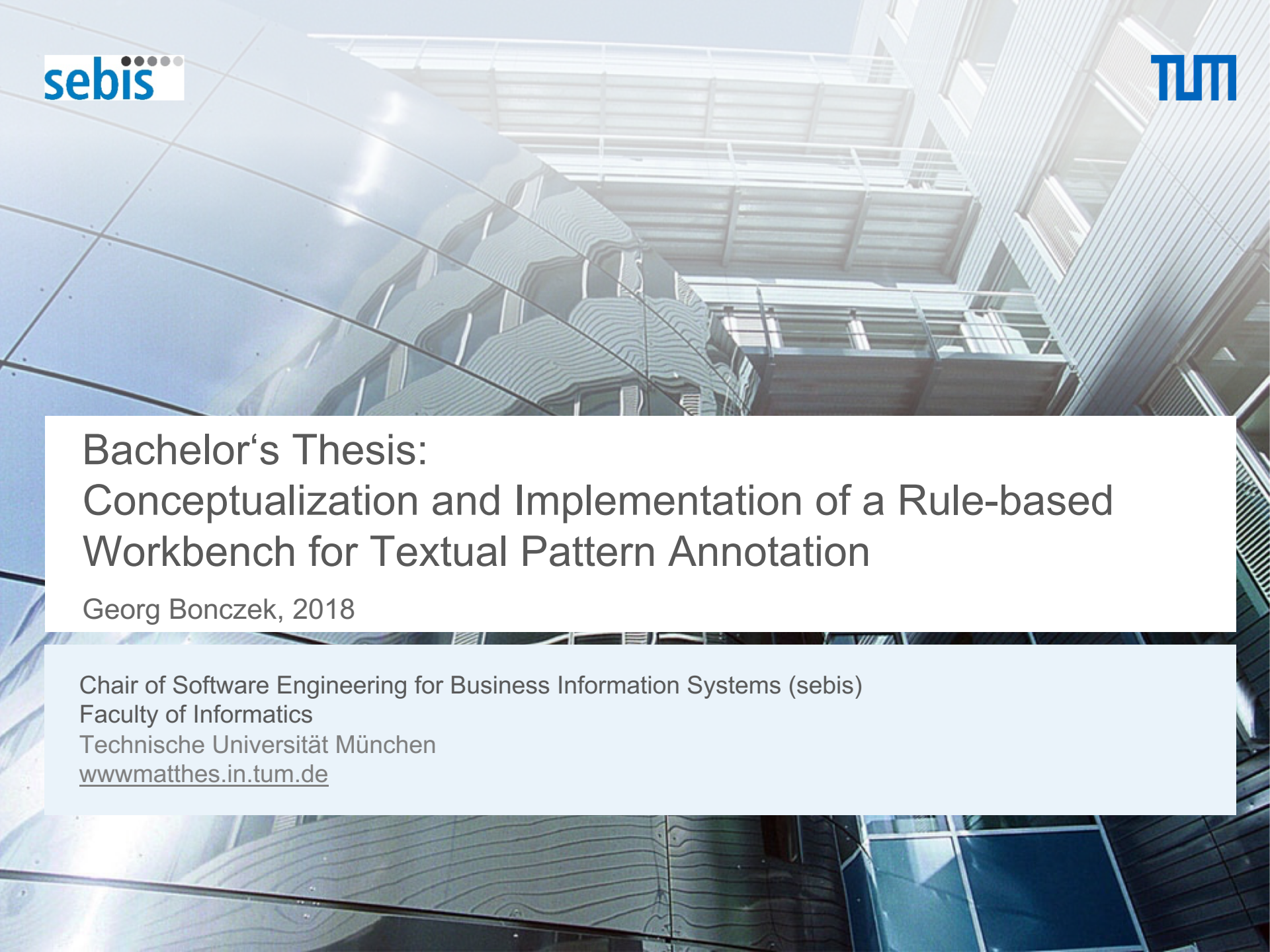


# Bachelor‘s Thesis: Conceptualization and Implementation of a Rule-based Workbench for Textual Pattern Annotation

Georg Bonczek, 2018

Chair of Software Engineering for Business Information Systems (sebis)
Faculty of Informatics
Technische Universität München
wwwmatthes.in.tum.de

# Motivation

Rule-based text annotation is still useful in times of machine learning:

- De-Facto Industry standard for information extraction [1]
- Easy and fast to implement
- Incorporation of domain knowledge

# Motivation

Rule-based text annotation is still useful in times of machine learning:

- De-Facto Industry standard for information extraction [1]
- Easy and fast to implement
- Incorporation of domain knowledge

Domain experts often require support of software engineers

- Goal: **Reduction of overhead in such a cooperation**

# Research Questions

- Which advantages and disadvantages do rule-based approaches have?
- How does a typical workflow for rule engineering in the legal domain look like and which roles are involved?
- What is the current state of the art tool support for rule development?
- How can the barrier to the development of rules be lowered?

# Research Questions

- Which advantages and disadvantages do rule-based approaches have?
- How does a typical workflow for rule engineering in the legal domain look like and which roles are involved?
- What is the current state of the art tool support for rule development?
- How can the barrier to the development of rules be lowered?

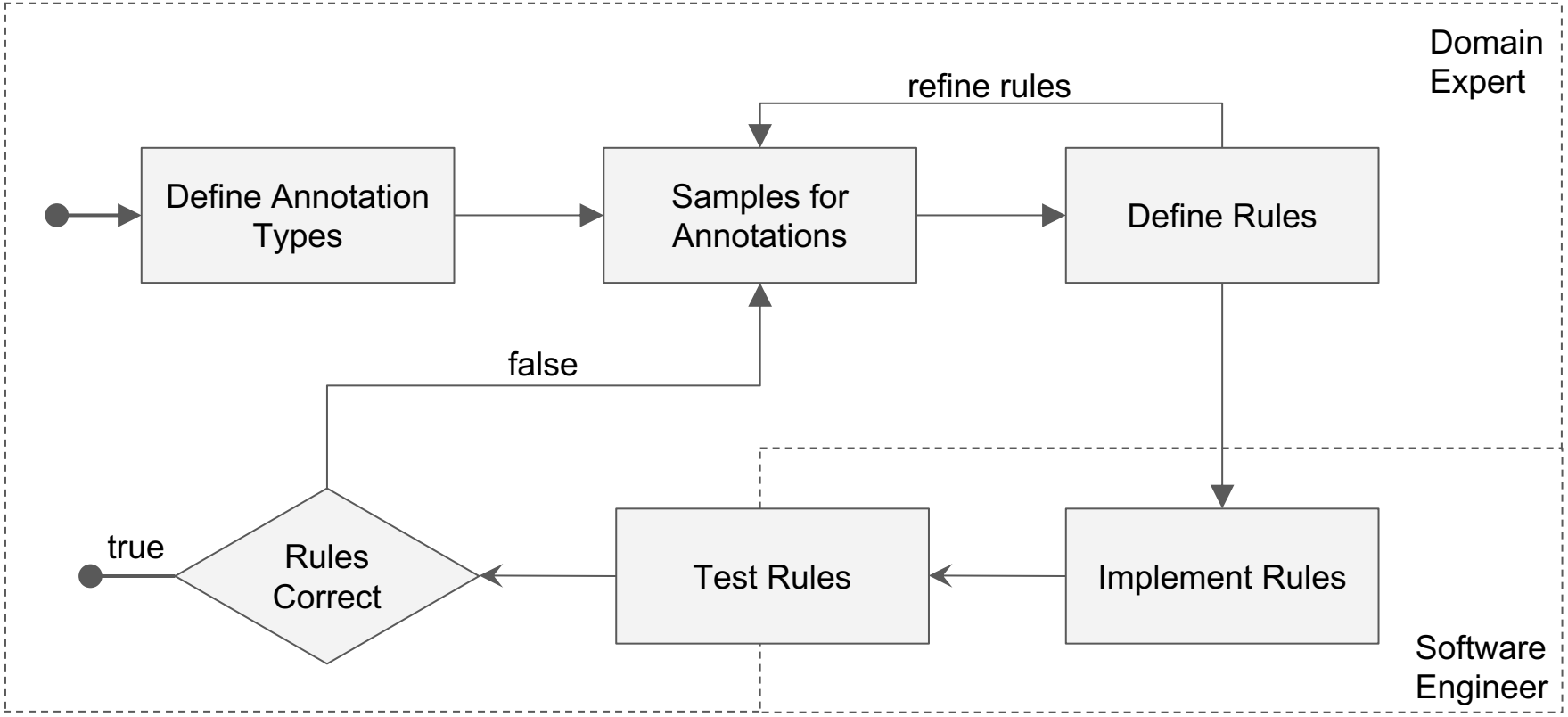
# Research Questions

- Which advantages and disadvantages do rule-based approaches have?
- How does a typical workflow for rule engineering in the legal domain look like and which roles are involved?
- What is the current state of the art tool support for rule development?
- How can the barrier to the development of rules be lowered?

# Research Questions

- Which advantages and disadvantages do rule-based approaches have?
- How does a typical workflow for rule engineering in the legal domain look like and which roles are involved?
- What is the current state of the art tool support for rule development?
- How can the barrier to the development of rules be lowered?

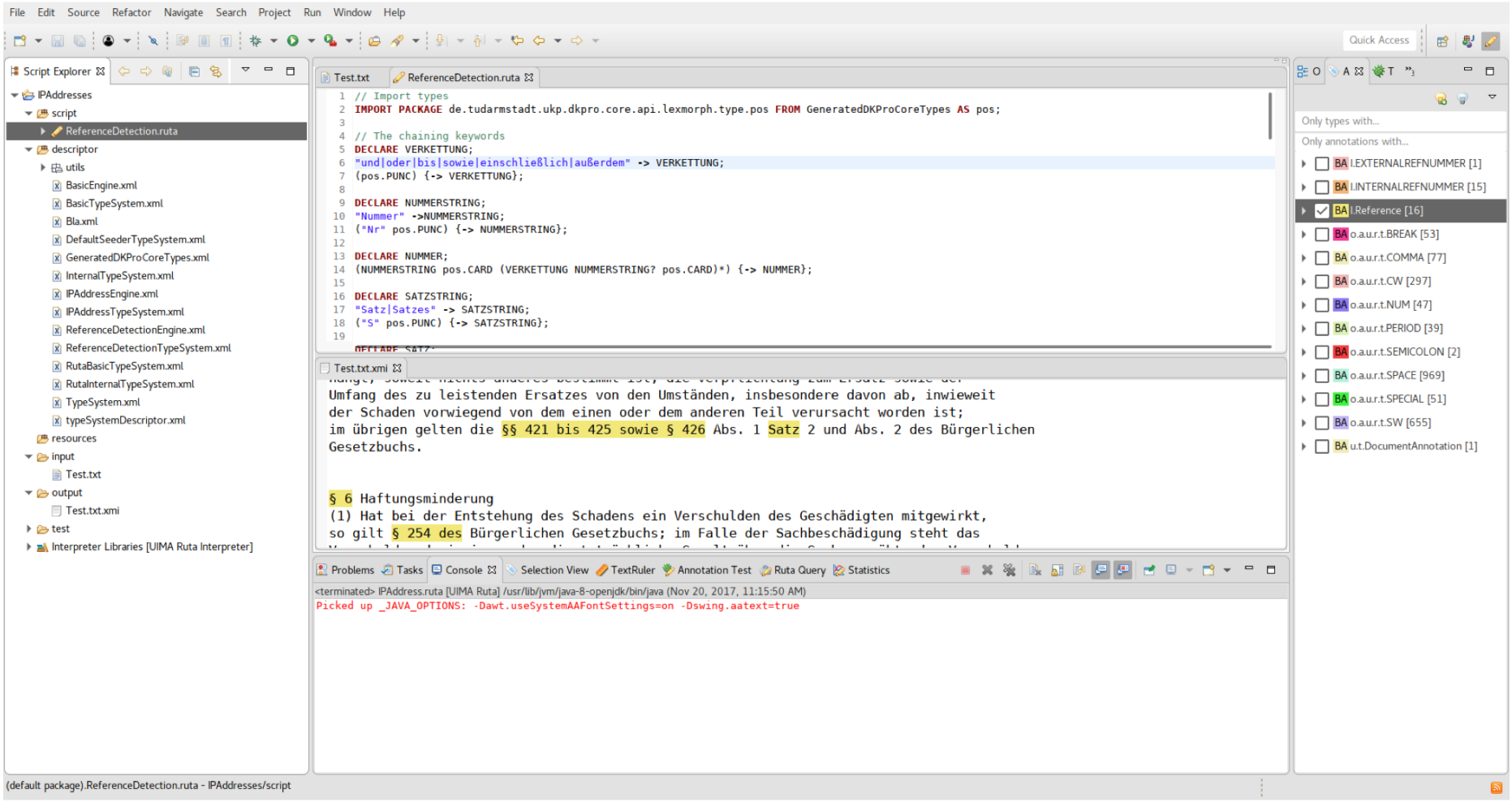
# Previous Workflow



# Previous Obstacles

- Rule languages are not targeted at non-technical users
- Conceptualization, Development and Evaluation phases are fragmented
- Necessary collaboration made difficult
- IDEs are code-centered not document-centered

# UIMA Ruta Workbench



# Approach

- Rule languages are not targeted at non-technical users
- **Minimal, extensible rule expression language**
- Conceptualization, Development and Evaluation phases are fragmented
- IDEs are code-centered not document-centered
- Necessary collaboration made difficult

# UIMA Ruta

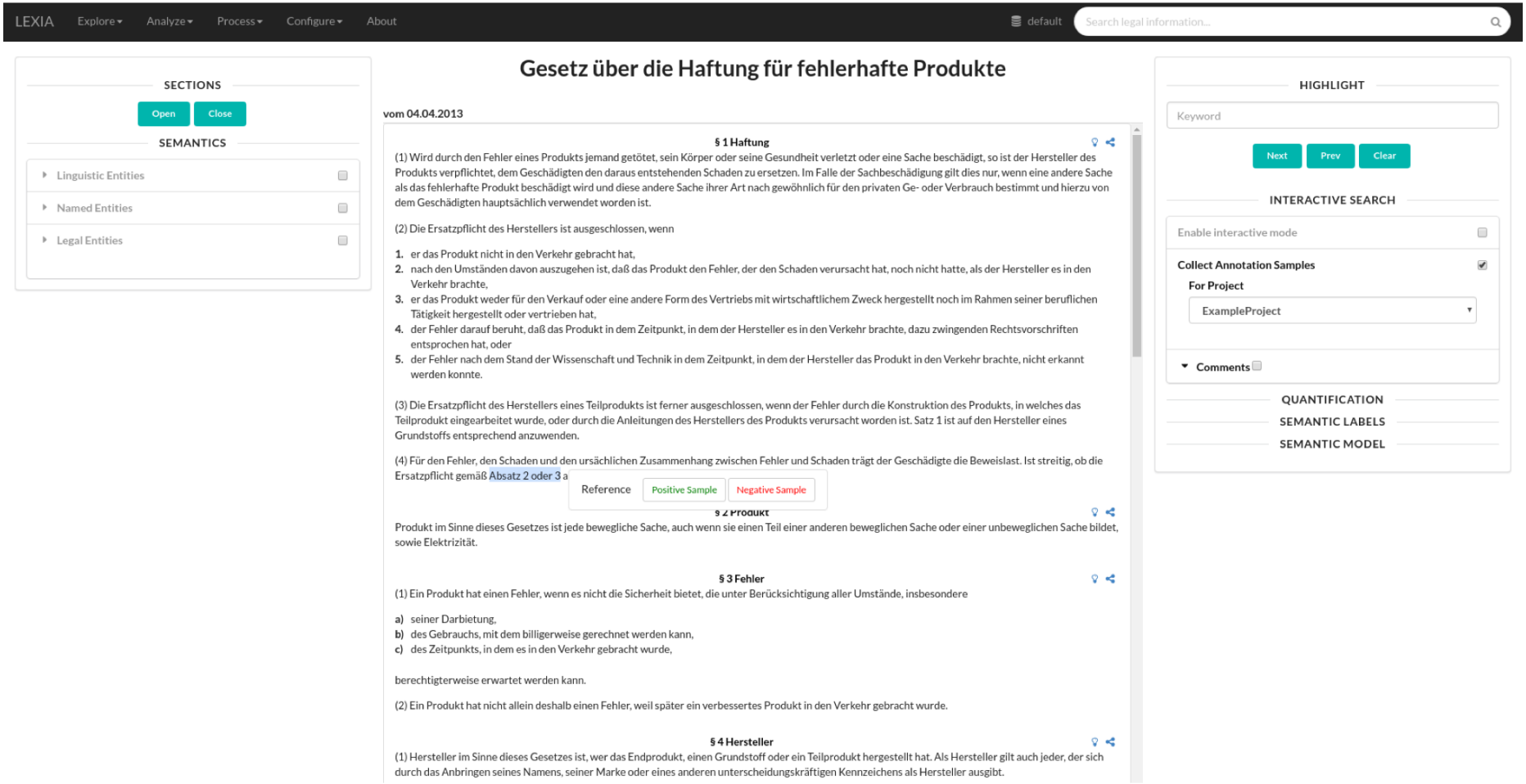
```
DECLARE INFINITIV;
V.PosValue == "VVIZU" {-> INFINITIV};
V.PosValue == "VAINF" {-> INFINITIV};
V.PosValue == "VNINF" {-> INFINITIV};
V.PosValue == "VVINF" {-> INFINITIV};

DECLARE ISTINFINITV;
(W{REGEXP("ist|sind")} # W{REGEXP("zu")} INFINITIV) {->ISTINFINITV};
(W{REGEXP("hat|haben")} # W{REGEXP("zu")} INFINITIV) {->ISTINFINITV};
(W{REGEXP("ist|sind")} # V.PosValue == "VVIZU") {->ISTINFINITV};
(W{REGEXP("hat|haben")} # V.PosValue == "VVIZU") {->ISTINFINITV};

INFINITIV {-> UNMARK(INFINITIV)}
ISTINFINITIV {-> UNMARK(ISTINFINITIV)}
```

# Raft

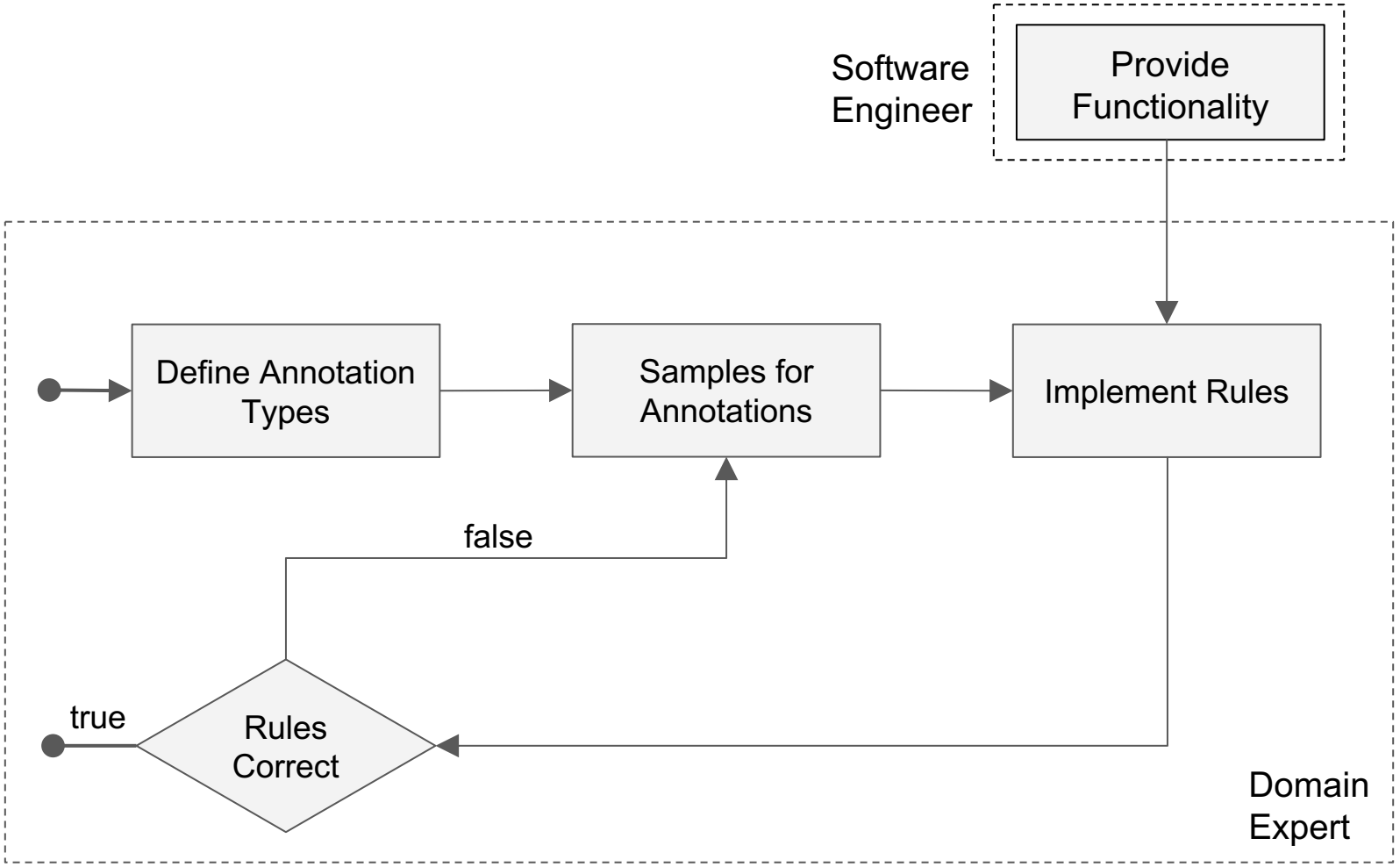
```
tmp INFINITIV
V{isInfinitive()} -> INFINITIV


tmp ISTINFINITV
"ist|sind" & "zu" INFINITIV -> ISTINFINITV
"hat|haben" & zu" INFINITIV -> ISTINFINITV
"ist|sind" & V{isZuInfinitive()} -> ISTINFINITV
"hat|haben" & V{isZuInfinitive()} -> ISTINFINITV
```

# Approach

- Rule languages are not targeted at non-technical users
- Conceptualization, Development and Evaluation phases are fragmented
- Necessary collaboration made difficult
- IDEs are code-centered not document-centered
- **Create and merge necessary tools into Lexia**


LEXIA
Explore
Analyze
Process
Configure
About
default
Search legal information...
SECTIONS
Open
Close
SEMANTICS
Linguistic Entities
Named Entities
Legal Entities
Gesetz über die Haftung für fehlerhafte Produkte
vom 04.04.2013
§ 1 Haftung
(1) Wird durch den Fehler eines Produkts jemand getötet, sein Körper oder seine Gesundheit verletzt oder eine Sache beschädigt, so ist der Hersteller des Produkts verpflichtet, dem Geschädigten den daraus entstehenden Schaden zu ersetzen. Im Falle der Sachbeschädigung gilt dies nur, wenn eine andere Sache als das fehlerhafte Produkt beschädigt wird und diese andere Sache ihrer Art nach gewöhnlich für den privaten Ge- oder Verbrauch bestimmt und hierzu von dem Geschädigten hauptsächlich verwendet worden ist.
(2) Die Ersatzpflicht des Herstellers ist ausgeschlossen, wenn
1. er das Produkt nicht in den Verkehr gebracht hat,
2. nach den Umständen davon auszugehen ist, daß das Produkt den Fehler, der den Schaden verursacht hat, noch nicht hatte, als der Hersteller es in den Verkehr brachte,
3. er das Produkt weder für den Verkauf oder eine andere Form des Vertriebs mit wirtschaftlichem Zweck hergestellt noch im Rahmen seiner beruflichen Tätigkeit hergestellt oder vertrieben hat,
4. der Fehler darauf beruht, daß das Produkt in dem Zeitpunkt, in dem der Hersteller es in den Verkehr brachte, dazu zwingenden Rechtsvorschriften entsprochen hat, oder
5. der Fehler nach dem Stand der Wissenschaft und Technik in dem Zeitpunkt, in dem der Hersteller das Produkt in den Verkehr brachte, nicht erkannt werden konnte.
(3) Die Ersatzpflicht des Herstellers eines Teilprodukts ist ferner ausgeschlossen, wenn der Fehler durch die Konstruktion des Produkts, in welches das Teilprodukt eingearbeitet wurde, oder durch die Anleitungen des Herstellers des Produkts verursacht worden ist. Satz 1 ist auf den Hersteller eines Grundstoffs entsprechend anzuwenden.
(4) Für den Fehler, den Schaden und den ursächlichen Zusammenhang zwischen Fehler und Schaden trägt der Geschädigte die Beweislast. Ist streitig, ob die Ersatzpflicht gemäß Absatz 2 oder 3 a
Reference
Positive Sample
Negative Sample
§ 2 Produkt
Produkt im Sinne dieses Gesetzes ist jede bewegliche Sache, auch wenn sie einen Teil einer anderen beweglichen Sache oder einer unbeweglichen Sache bildet, sowie Elektrizität.
§ 3 Fehler
(1) Ein Produkt hat einen Fehler, wenn es nicht die Sicherheit bietet, die unter Berücksichtigung aller Umstände, insbesondere
a) seiner Darbietung,
b) des Gebrauchs, mit dem billigerweise gerechnet werden kann,
c) des Zeitpunkts, in dem es in den Verkehr gebracht wurde,
berechtigterweise erwartet werden kann.
(2) Ein Produkt hat nicht allein deshalb einen Fehler, weil später ein verbessertes Produkt in den Verkehr gebracht wurde.
§ 4 Hersteller
(1) Hersteller im Sinne dieses Gesetzes ist, wer das Endprodukt, einen Grundstoff oder ein Teilprodukt hergestellt hat. Als Hersteller gilt auch jeder, der sich durch das Anbringen seines Namens, seiner Marke oder eines anderen unterscheidungskräftigen Kennzeichens als Hersteller ausgibt.
HIGHLIGHT
Keyword
Next
Prev
Clear
INTERACTIVE SEARCH
Enable interactive mode
Collect Annotation Samples
For Project
ExampleProject
Comments
QUANTIFICATION
SEMANTIC LABELS
SEMANTIC MODEL

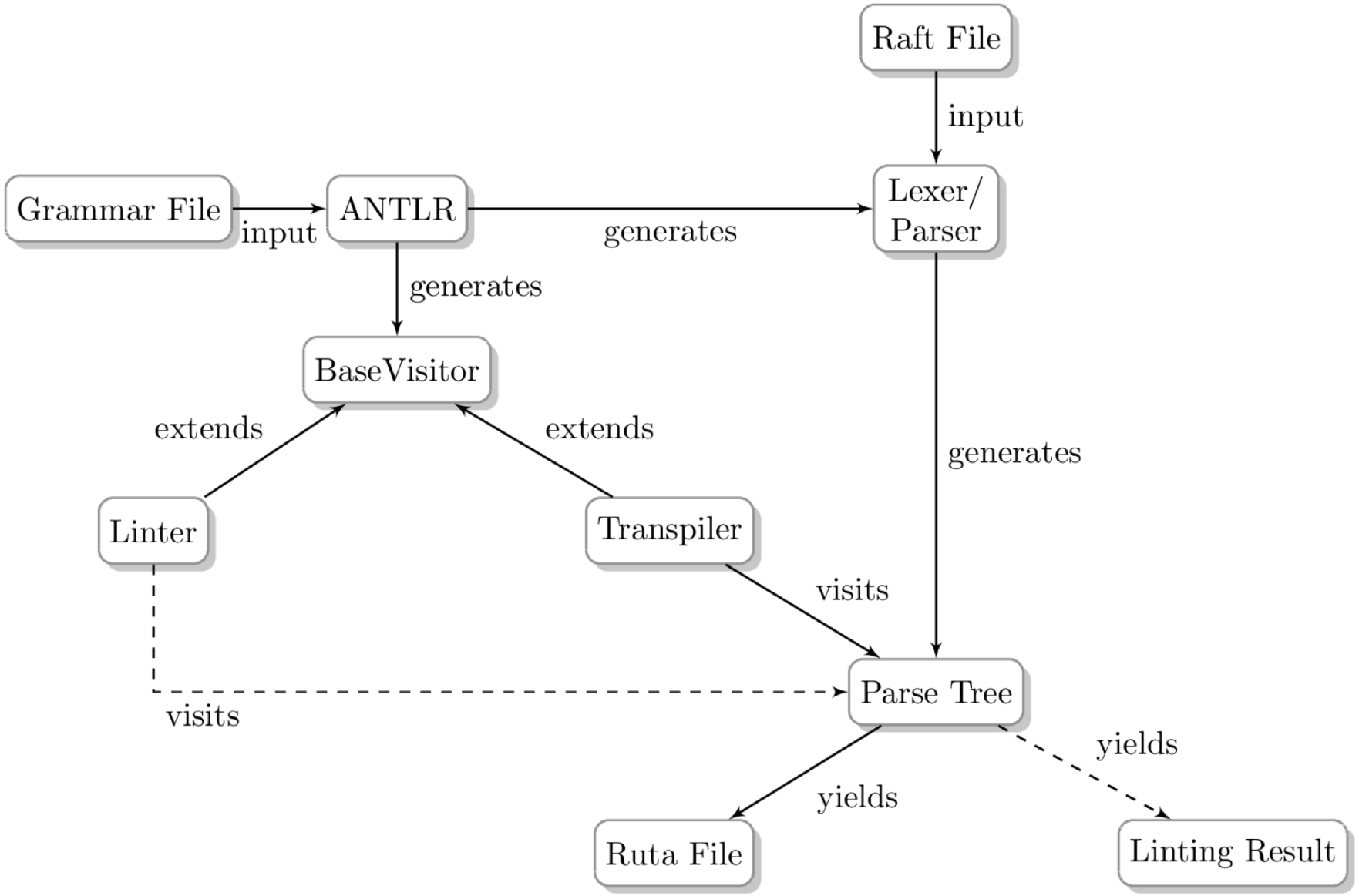
# Current Workflow



# Rule Language

- DSL for linguistic phrase matching on legal texts
- Limited syntax and expressiveness by design
- Based on common code patterns in UIMA Ruta
- Extensible with Java
- Transpiled into UIMA Ruta

# Rule Language Implementation



# Demo

# Evaluation

- Feature comparison concerning workflow
- Extension of the Lexia workbench with integrated development workflow
- Code metrics
- Drastic reduction in source lines of code through DSLs

# Conclusion

- Conventional tools suffer in interdisciplinary contexts
- Most of these drawbacks can be mitigated
- DSLs as powerful pier of the workflow
- Realizing potential of rule-based information extraction

# Questions

[1] Chiticariu, Laura, Yunyao Li, and Frederick R. Reiss. "Rule-based information extraction is dead! long live rule-based information extraction systems!." Proceedings of the 2013 conference on empirical methods in natural language processing. 2013.

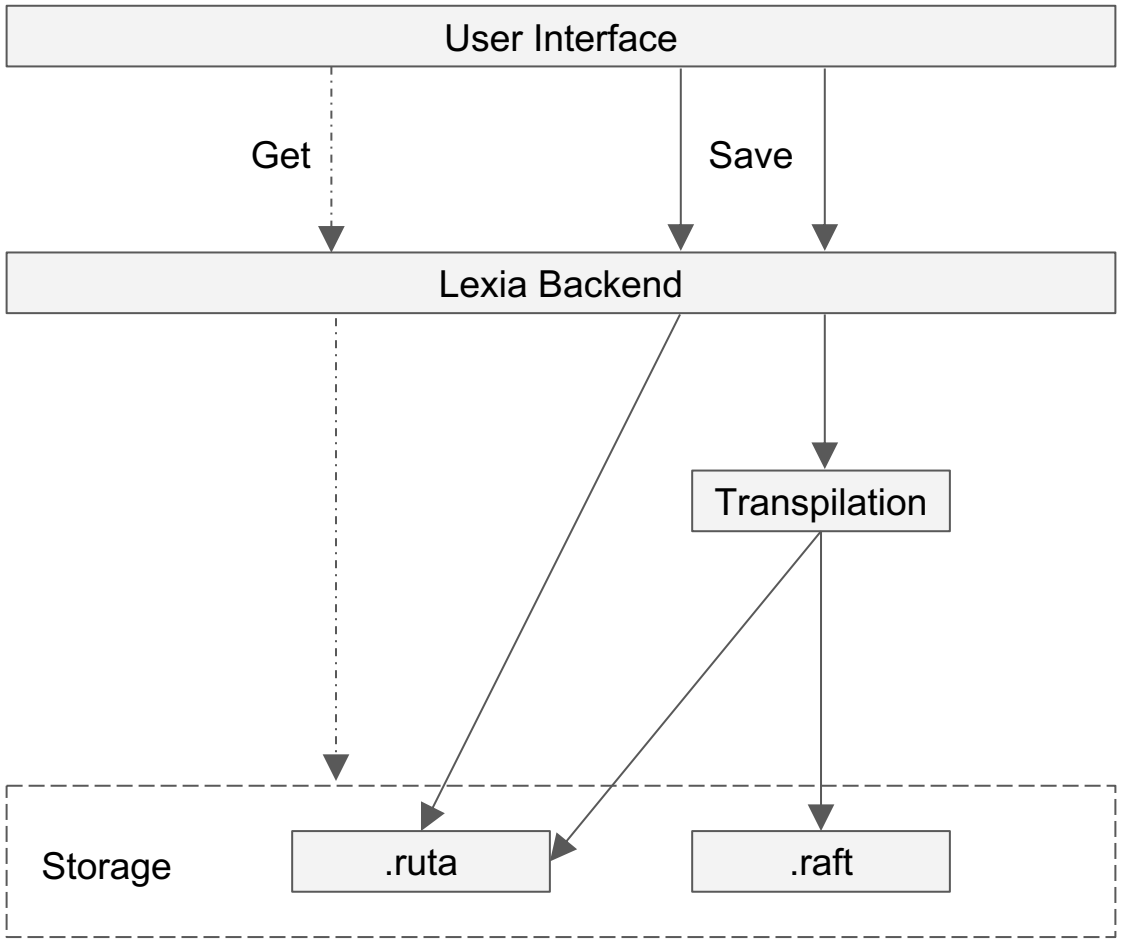
[2] J. Pustejovsky and A. Stubbs. Natural Language Annotation for Machine Learning. 2013.

# Rule Language Extensions

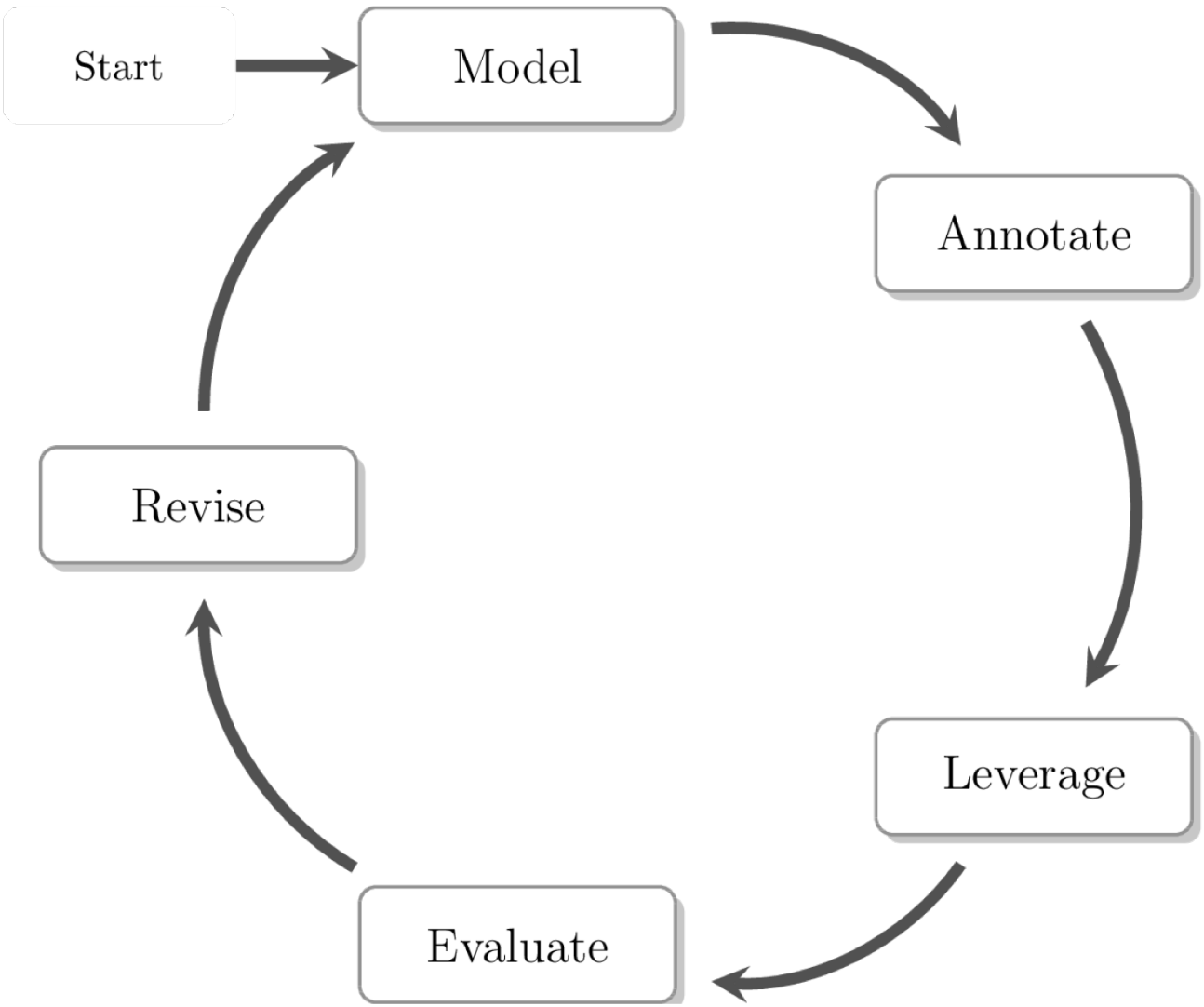
- Implemented using Java Annotations
- Automatic Generation of Ruta Extension classes
- Averts boilerplate of Ruta Extensions

```
/**
   * Tests if the provided token is an infinitive
   * @return true if the token is an infinitive
   */
  @RutaCondition(targetPackage = "lexia") public static boolean isInfinitive(RutaContext context) {
    Feature f =
        context.getAnnotation().getCAS().getAnnotationType().getFeatureByBaseName("morphTag");
    String fval = context.getAnnotation().getFeatureValueAsString(f);
    return fval.equals("VVIZU") || fval.equals("VAINF") || fval.equals("VNINF") || fval
        .equals("VVINF");
  }
```

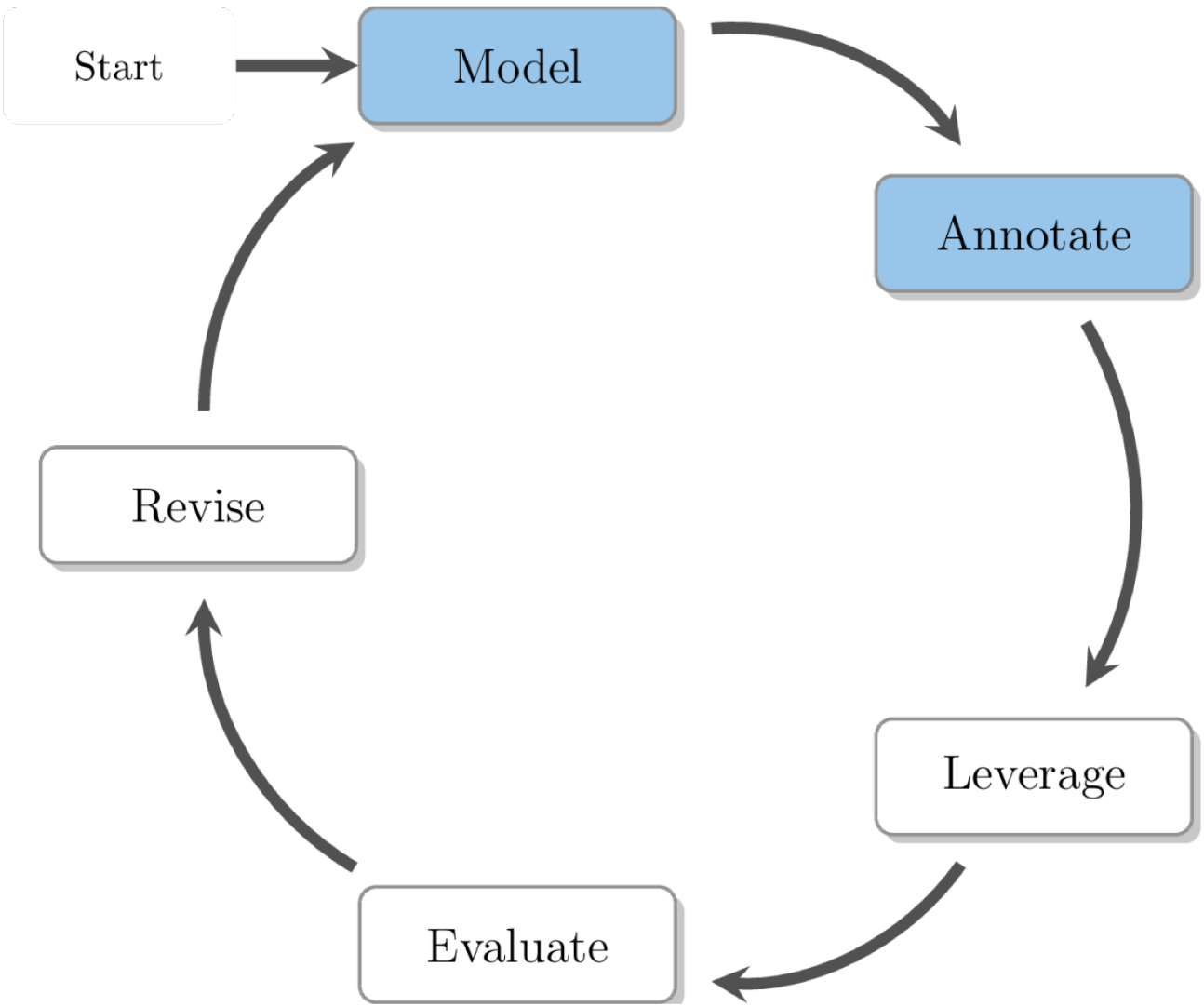
# Rule Language Integration



# Base Workflow



# Base Workflow



# Base Workflow