The Déductions project Architecture

J.M. Vanel

Summary

- Demonstration
- Artificial Intelligence
- Data flux (user)
- Top level sequence diagram
- The N3 language
- Rules data flow (implementation)
- EulerGUI IDE
- The Déductions project
- Conclusion

EulerGUI and Déductions Demonstration

- Download EulerGUI
- Java Swing application generator from OWL model and N3 logic rules:
  Deduction project How To

Demonstration

- http://eulergui.svn.sourceforge.net/viewvc/eulerg
- Modèle importé quelconque en OWL
- Montrer le modèle dans Protégé
- On choisit un point de départ (editedClass)
- On génère une application avec formulaires de saisie, et qui sauve au format N3.
- Fonctionalités avancées dans les champs de saisie
- Montrer une règle de validation en N3

AI

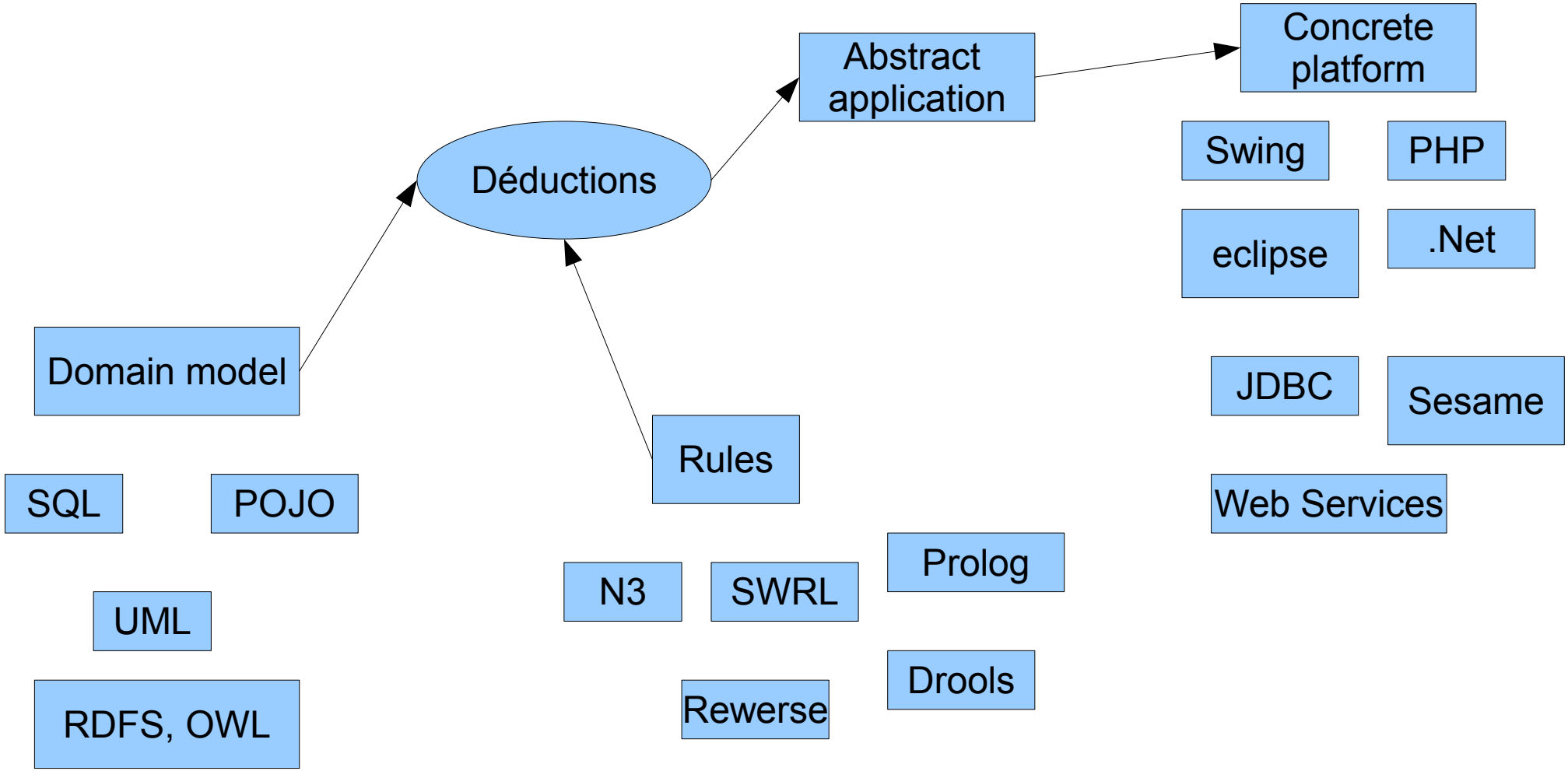
- Small data size => larger rule base possible
- Choose the right engine (Drools)
- Choose the right file format : N3
-
- FOL, Description Logic
- difference between RETE engine and full logic engine (wumpus example, disjunction in consequent)

OO and KB

- Object Oriented for the business data has lived
  - In OO, data model, bizz rules, infrastructure are mixed
- time for knowledge bases !
- OO remains fit for the infrastructure code though
- A Copernican revolution !

The data flux (user p.o.v.)

Re-use current technology



Let the Models Come to us

- SQL
- UML
- EMF
- Pojo (Plain Old Java Objects)
- XML Schema

Will have input connectors for all dialects.

Let the Ontologies Come to us

- RDF
- OWL
- KIF
- Classic
- TPTP
- ...

Will have input connectors for all dialects.

Let the Rules Come to us

- N3
- SPARQL (queries only)
- Drools
- Prolog
- SWRL, RIF, Rewerse, ...
- ...

Will have input connectors for all dialects.

But What is the Esperanto? N3

Why N3?

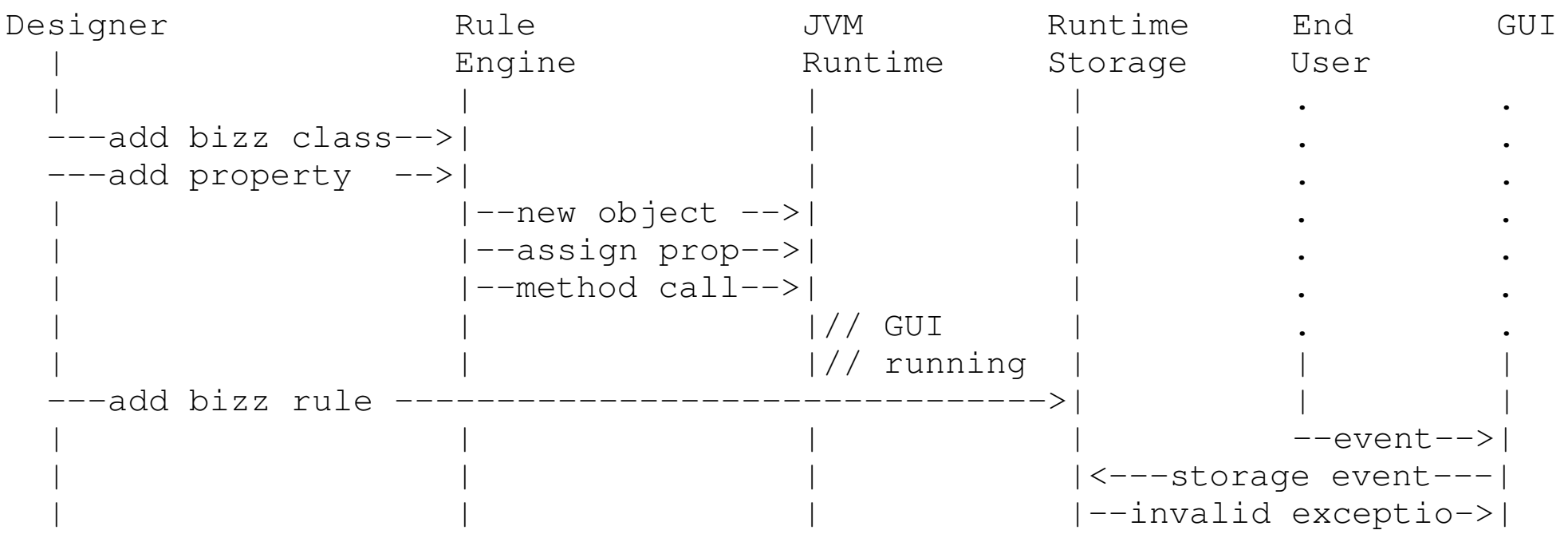
- can represent data, classes and properties, rules.
- can represent UML, XMI, and SQL and more
- naturally integrates RDF and OWL from W3C
- Introduction to N3 and RDF: http://www.w3.org/2000/10/swap/Primer
- tutorial introduction to N3 rules: http://www.w3.org/2000/10/swap/doc/Rules
- compare formats N3, SQL, UML,… http://www.w3.org/2000/10/swap/doc/formats

Panorama: the metamodel stack

W3C land	OMG land	AI		
N3 logic		FOL		
		Prolog		
OWL		Description Logic		
	UML			
RDFS	MOF, eCore			
RDF, N3	XMI			

- Expressivity is is higher up
- RDF can link anything

Top level sequence diagram



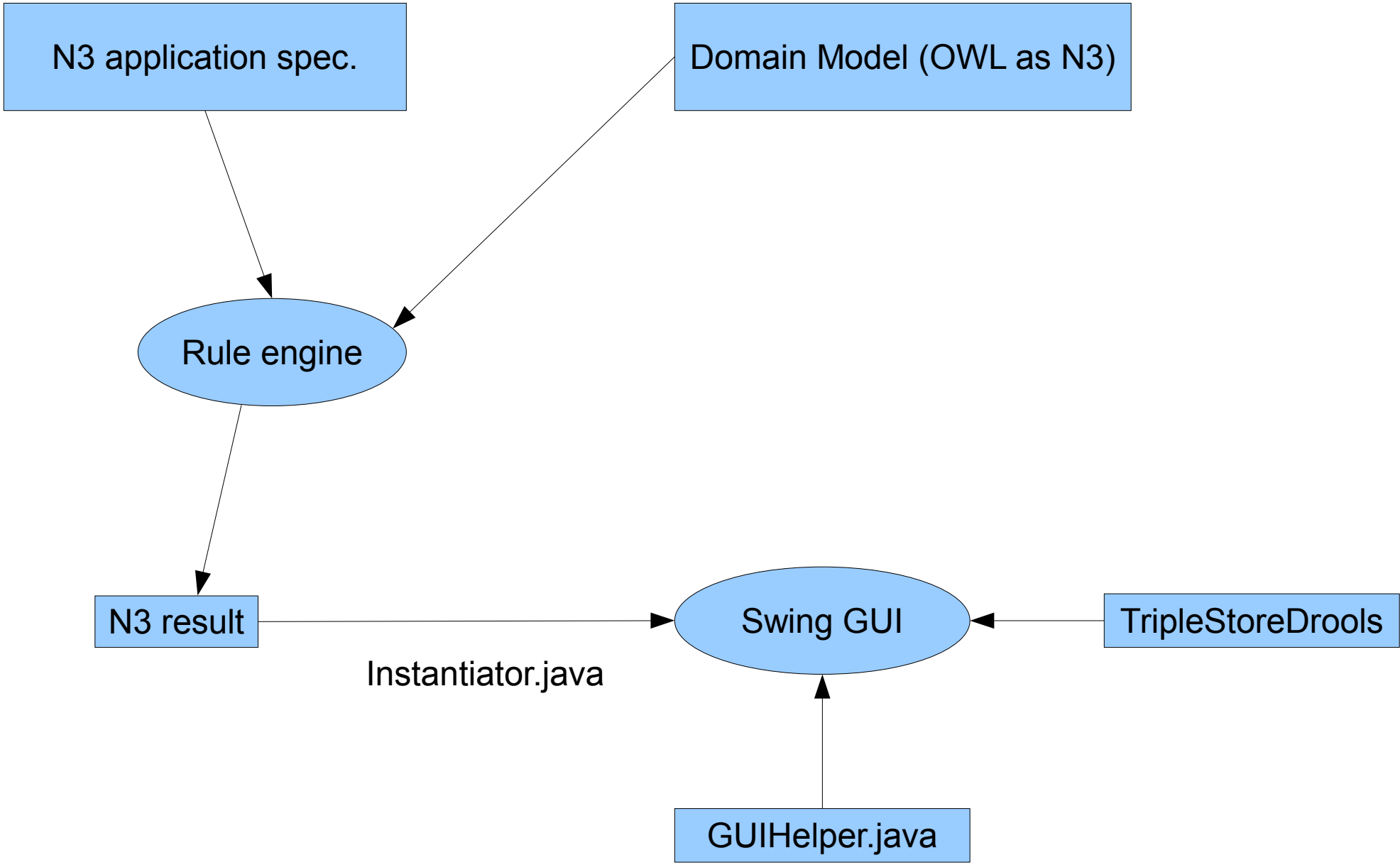
Notes:

- Runtime Storage is also a Rule Engine
- generated GUI possibly includes multi-view for the underlying bizz objects, including the update logic

Design time Rule Engine

- Designer enters/imports:
  - a domain model (classes and properties)
  - Business rules
  - A small application specification
- The rest is background knowledge (supporting ontologies and rules)

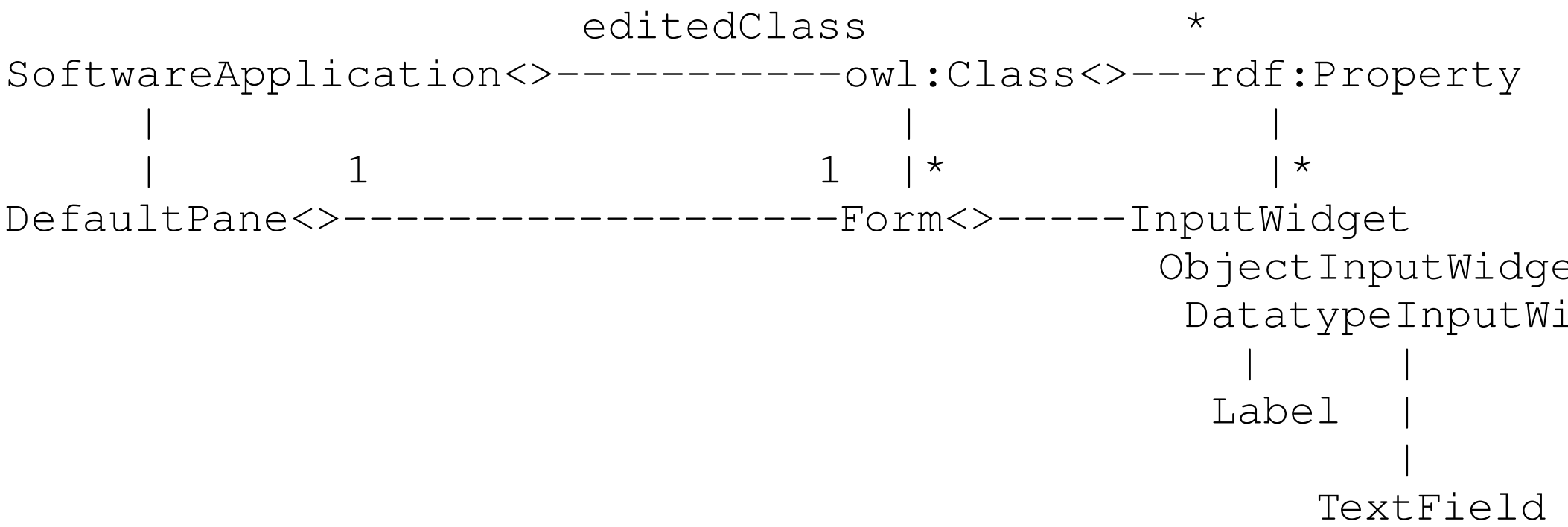
Current rules data flow



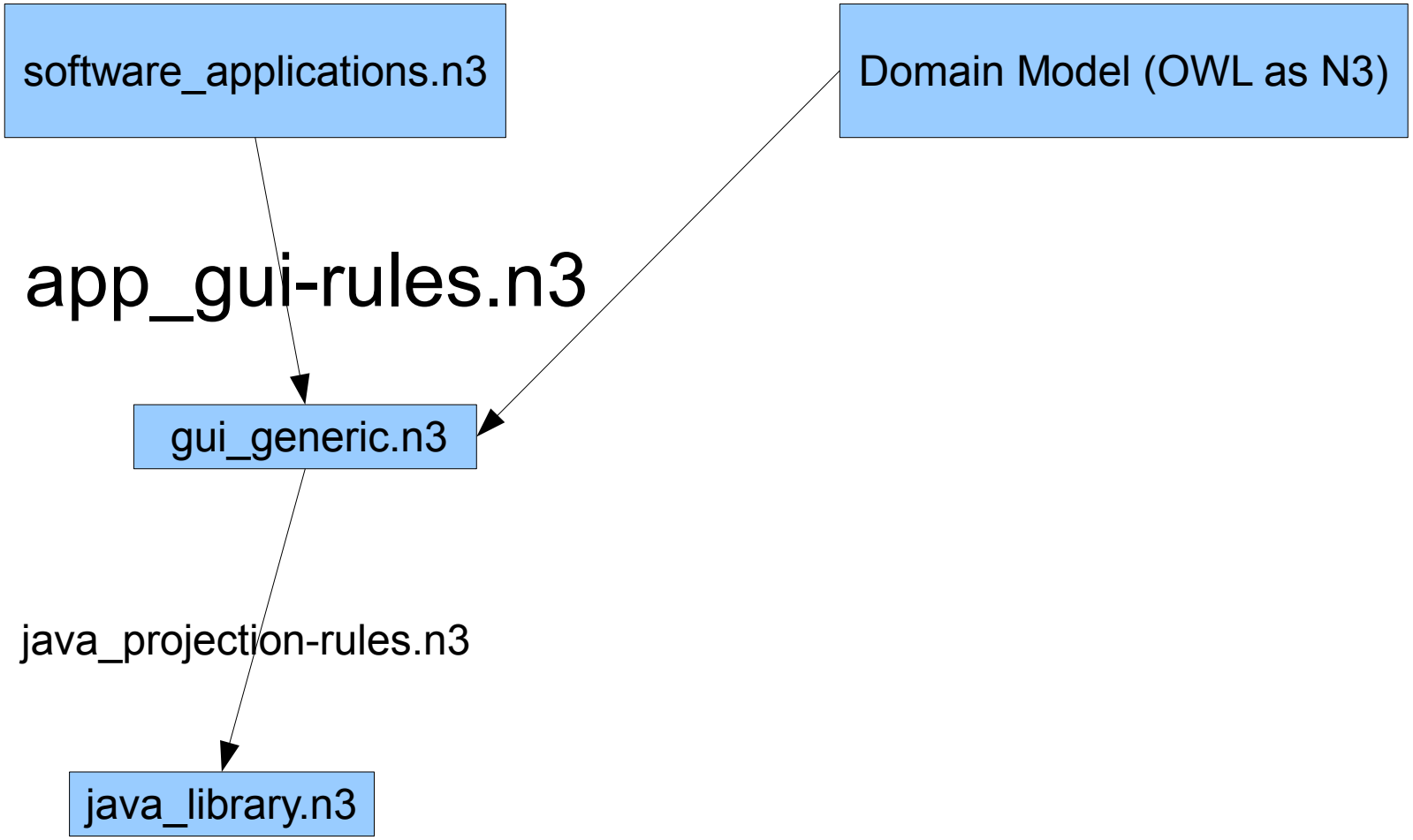
Supporting ontologies stack

- software applications
- business applications TODO
- generic GUI: widget/component, callback/action; this is a high-level model describing the GUI from the user point of view
- user interactions: user actions: create/update/delete, query/view, user goals/intentionsstill sketchy; not yet used
- abstract convergence platform; software application, generic GUI, etc, are to be translated in terms of it - TODO
- concrete platforms (Java SE, Java J2EE, SWI Prolog + XPCE, Python, PHP, Ruby on Rails, ...); this is the low level layer
- software purposes (point to most human activities being kind of management or viewing/search/navigation) ; for now 2 properties in software applications
- software project (software development) : classes(ontology), rules, purpose, platform, libraries, deployment, human roles, team, version, realease, test, specification, documentation, plus notions covered by UML 1 and 2 - TODO

Application-->GUI rules



Current rules flow



The blue rectangles bear the name of class model for the N3 data.

Rule example 0

```
# when C has a parent P, then C is the child of P :


{
   ?C :hasParent ?P .
} => {
  ?P :hasChild ?C .
} .
```

?C, ?P
are universally qualified

Rule example 1

```
# add a field in the form for each property of a
   class:
{  ?CLASS gui:hasForm ?FORM .
   ?PROP rdfs:domain ?CLASS .
} => {
  ?FORM gui:hasField ?FIELD .
  ?FIELD gui:inputWidgetSpecification ?PROP .
} .
```

?CLASS, ?FORM, ?PROP are universally qualified

FIELD is existentially qualified

Rule example 2

```
# the type of the field depends on the type of the
   property: ObjectProperty or DatatypeProperty
{
  ?FIELD gui:inputWidgetSpecification ?PROP .
  ?PROP a owl:DatatypeProperty .
} => {
  ?FIELD a gui:DatatypeInputWidget .
} .
```

OWL implemented with N3 logic

- as part of the Euler project, a library of N3 rules implements the logic of OWL and RDF Schema (transitive property, inheritance, etc), and other goodies, see:
- http://eulersharp.svn.sourceforge.net/viewvc/eul
-
- {?P a owl:TransitiveProperty.
- ?S ?P ?X.
- ?X ?P ?O. } => { ?S ?P ?O }.

Euler GUI - Use cases

Open any number of RDF / OWL / N3 documents

- test and debug the rules using the 3 rule engines (Drools, Euler, CWM)
- Generate an application (Déductions framework)
- export all project as :
  - a set of Drools packages, plus the facts in XMLEncoder format
  - A command line for the Prolog engine

The Deductions project

- Application generation
  - platform independence
- User-friendliness : the Good Servant
- component-based application building: Intelligent modularity
- Comprehension without prior protocol

Advanced GUI features

- GUI rules: building components tree, behavior: cardinalities, inheritance, constraints (solve to infer values),
- Advanced features: propagate edits or not (money Xfer between accounts), has few values, graph view (following user past actions, lens), zip paradigm
- record user actions, and show some simple feed-back, maybe last object creations used for suggesting object link
- show table view (like relational DB table)
- show tree view : 1. follow object properties; 2. follow subclassOf , then rdf:type
- demonstrate UML front-end

GUI: the good servant

- every user action should be recorded
- exploit to infer her/his intentions
- Also draw all consequences from the model and data

Intelligent modularity : letting valences connect

- Re-use the wealth of existing libraries and components
- Tag libraries with their purpose
- Add protocol state machines
- Then we can infer actual call sequence and automate application building
- Also possibility to find libraries and applications by their functionalities

Comprehension without prior protocol

- Between human and computer
- Between computers
- Leverage on linguistics
- opencyc.org, WordNet, upper level ontologies: Sumo, Milo, ...
- ACE project (Attempto Controled English)

Conclusion

- Copernic revolution: the infrastructures and OO techniques are at the periphery, Ontologies and rules and at the center
- Reduce the Babel effect effect in computer science by applying AI techniques
- Automatize application building will allow IT projects to concentrate on essential matters: domain model and business rules