OO jDREW 1.0

General UI improvements and RuleML 1.0 knowledgebase support

by

Omar Alsaiari, Khalid F. AlMutariri, Nasser Albunian,

Christian Fabbricatore and Markus Zucker

November 21, 2011

UNIVERSITY OF NEW BRUNSWICK UNB

Agenda

1. Introduction and Motivation
2. Project infrastructure
3. GUI improvements
4. RuleML 0.91 & 1.0
5. Parser implementation
6. Test cases
7. Project demo
8. Conclusion and outlook

Introduction and Motivation

OO jDREW: is an object-oriented extension of the Java Deductive Reasoning Engine for the Web (jDREW)

Why OO jDREW: It is used for querying and processing facts and rules by either using Prolog, Positional and Slotted Language (POSL), or RuleML syntax.

Objectives (1)

Support for RuleML 1.0

- Identify changes from RuleML 1.0
- Rewrite parser infrastructure to support RuleML 1.0
- Differentiate between RuleML sub-languages

Project infrastructure

- Move code from SourceForge to GitHub
- Set up bug tracking and a proper project roadmap

Objectives (2)

Refactoring & Other features

- Code clean-up
- Graphical User Interface (GUI) improvements
- Definition of test cases

Document and test changes

New Project Infrastructure

- OO jDREW aims to be the Open Source reference implementation of RuleML
- No (public) version control or issue tracking
- Roadmap hidden in the RuleML wiki
- No real software development, just hacking
- Idea: Kick-start a new OSS project
  - Correctness
  - Understandability
  - Simplicity
  - Performance

New Project Infrastructure

- Version control
  - Changes should be comprehensible
  - Lack of version control leads to “Cargo-Cult Programming”
  - Enable collaboration
- Tool of choice: Distributed VCS
  - Import existing SVN repository into Git
  - Hosting platform: GitHub
    - Free for Open Source projects
    - JavaDoc hosting
    - Integrated issue management (bugtracking)

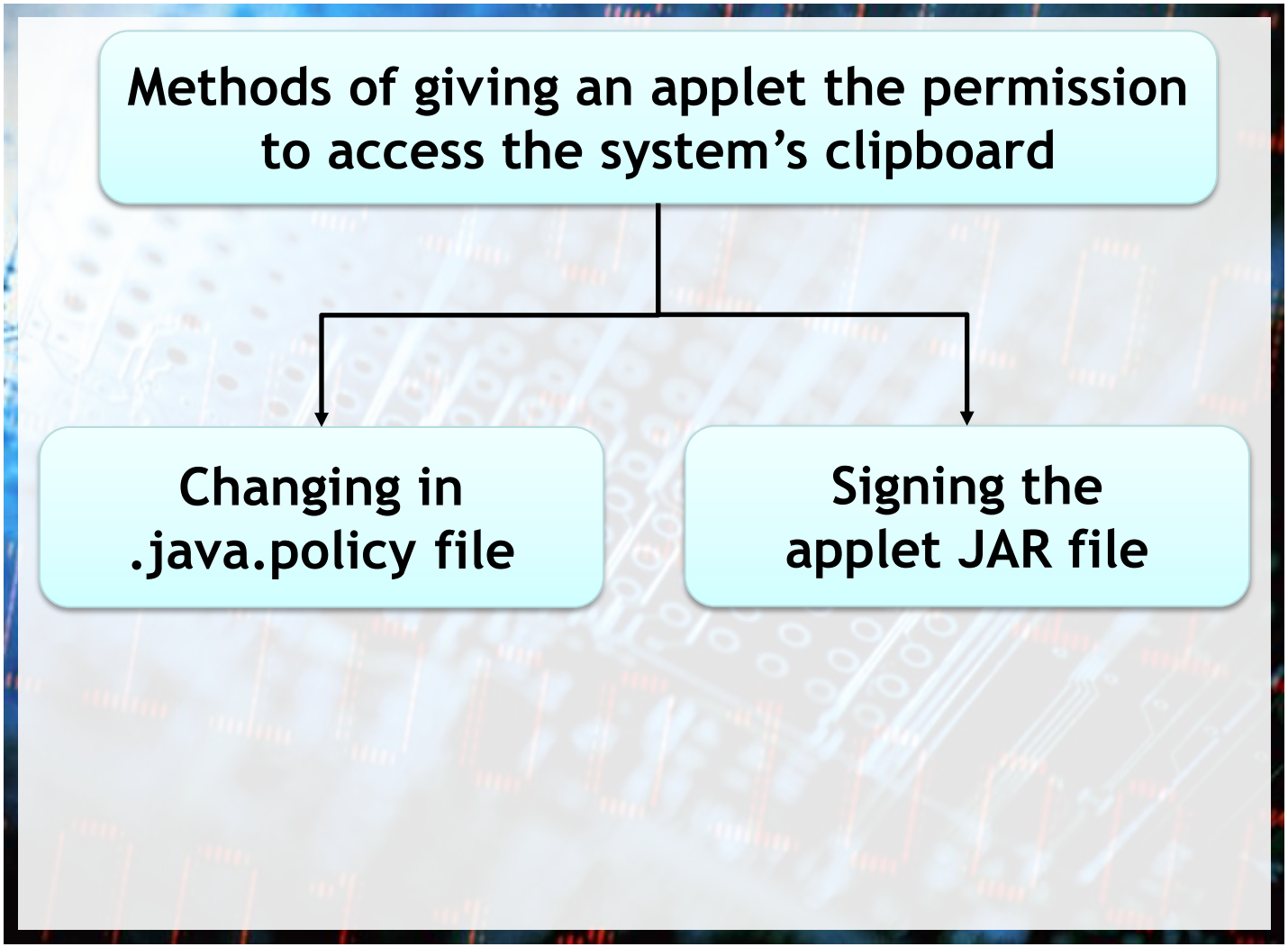
New Project Infrastructure

- More documented history in the past two weeks than in the last three years
- Next problem: dependency management, unit testing, packaging
  - Best available tool: Maven
    - Handles almost all project dependencies
    - Packaging of signed JARs comes for free
    - Test automation
    - Software quality reports

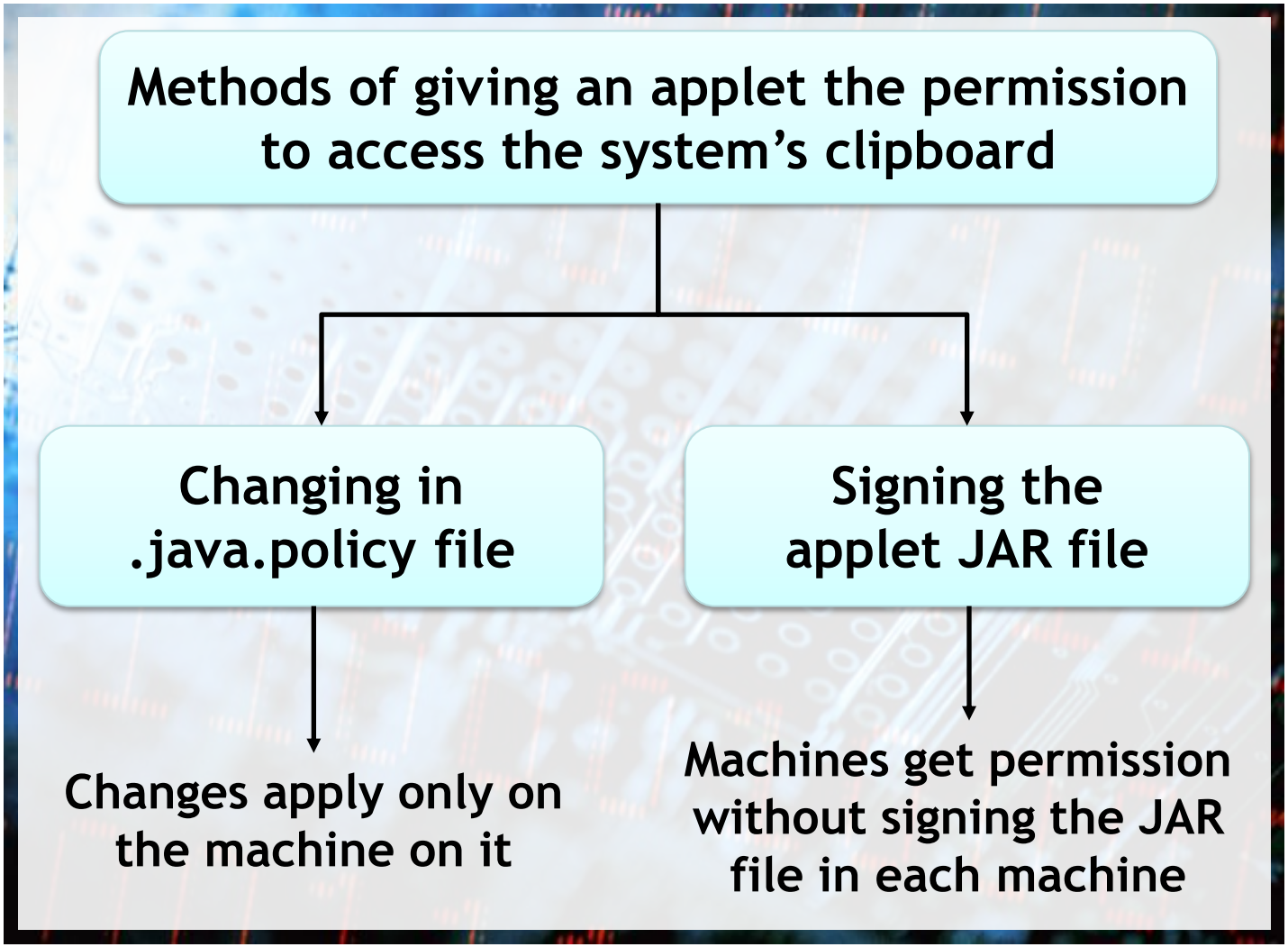
GUI (clipboard access / copy & paste)

- Users were not able to copy from web browsers, documents or applications to OO jDREW
- Java applets downloaded from the Internet or any remote source is untrusted, so that it must be restricted to access the system
- To allow applets to access the system clipboard, they must be given a explicit permission

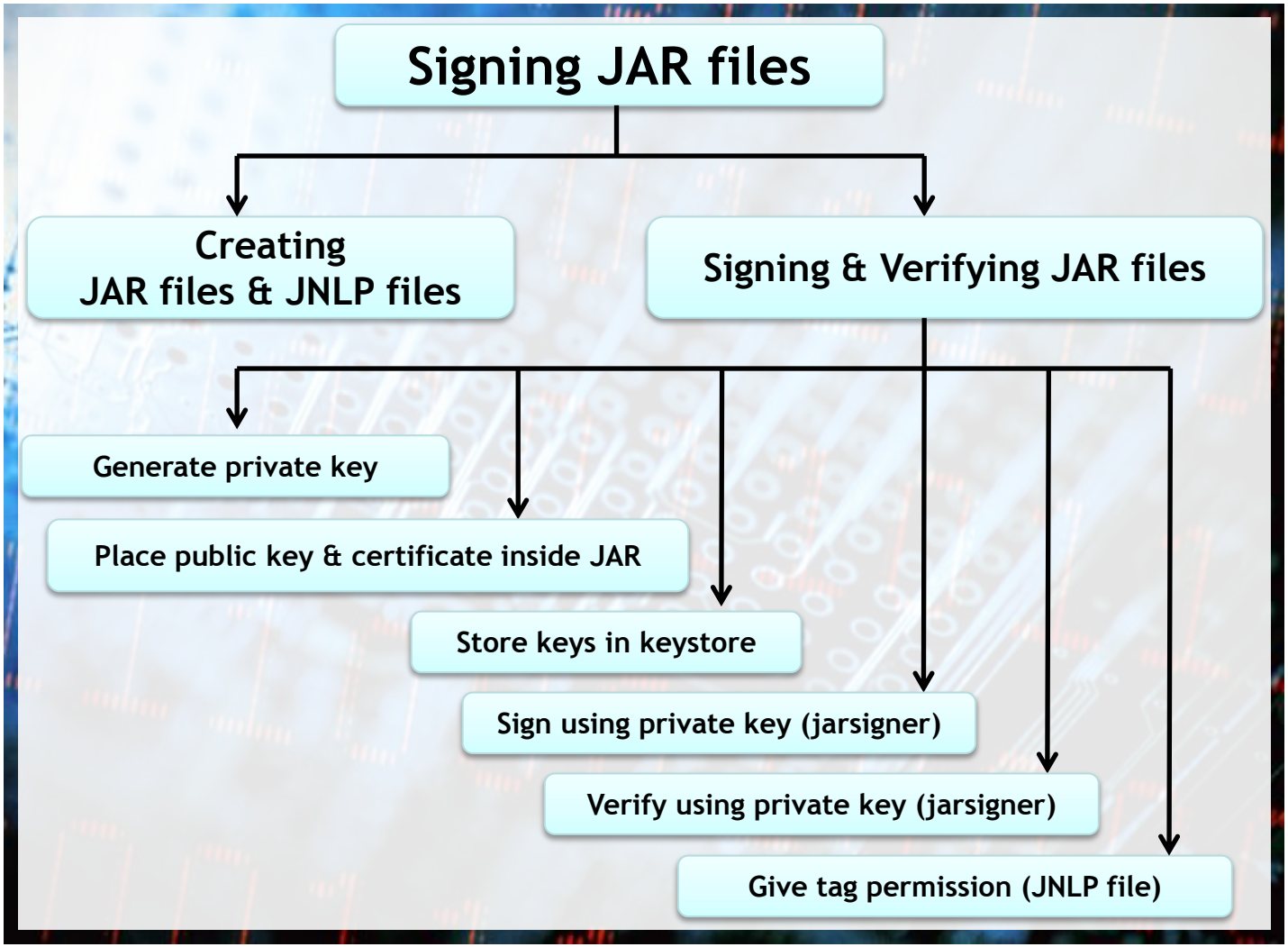
GUI - clipboard access / copy & paste (1)



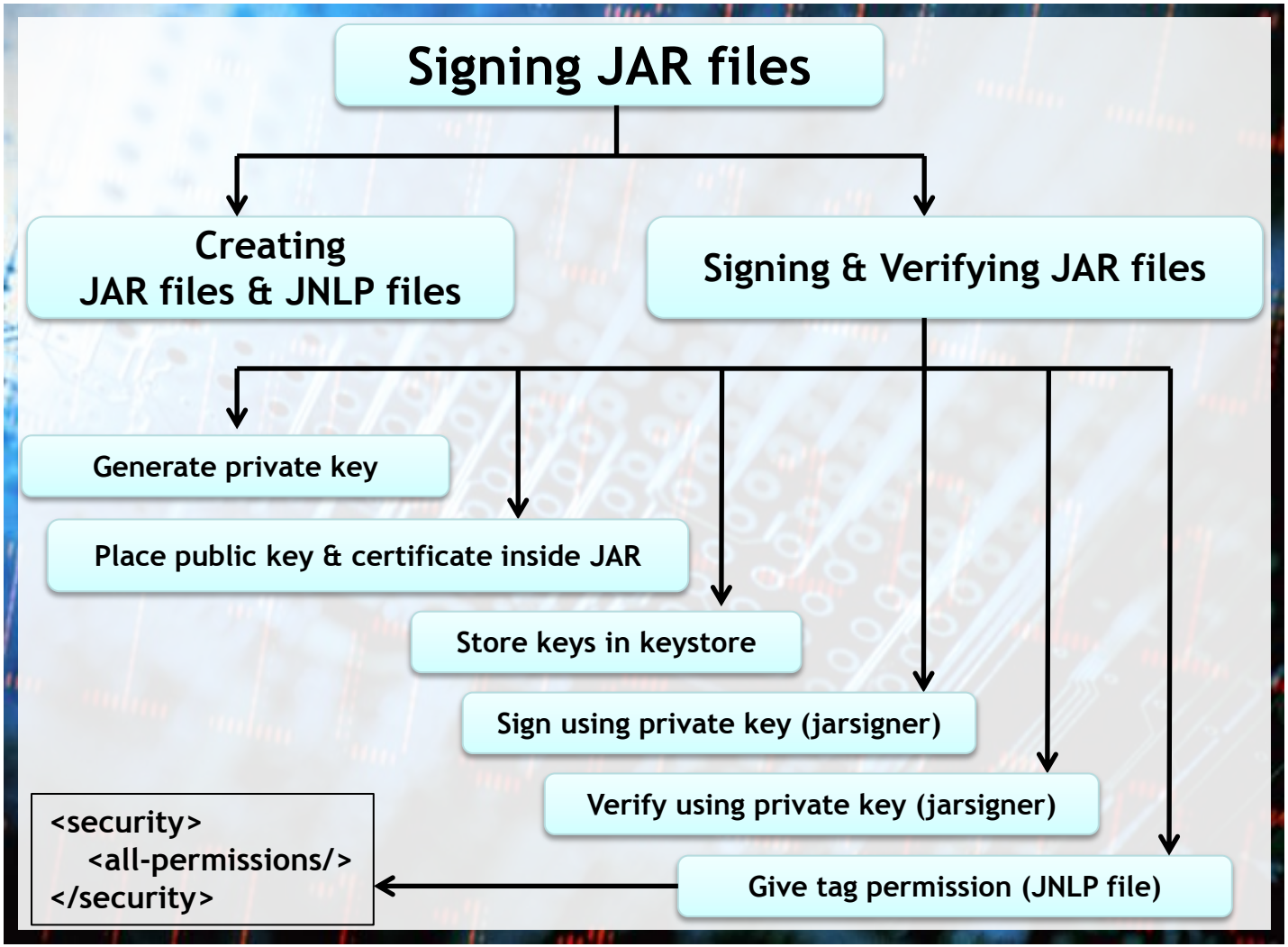
GUI - clipboard access / copy & paste (2)



GUI - clipboard access / copy & paste (2)

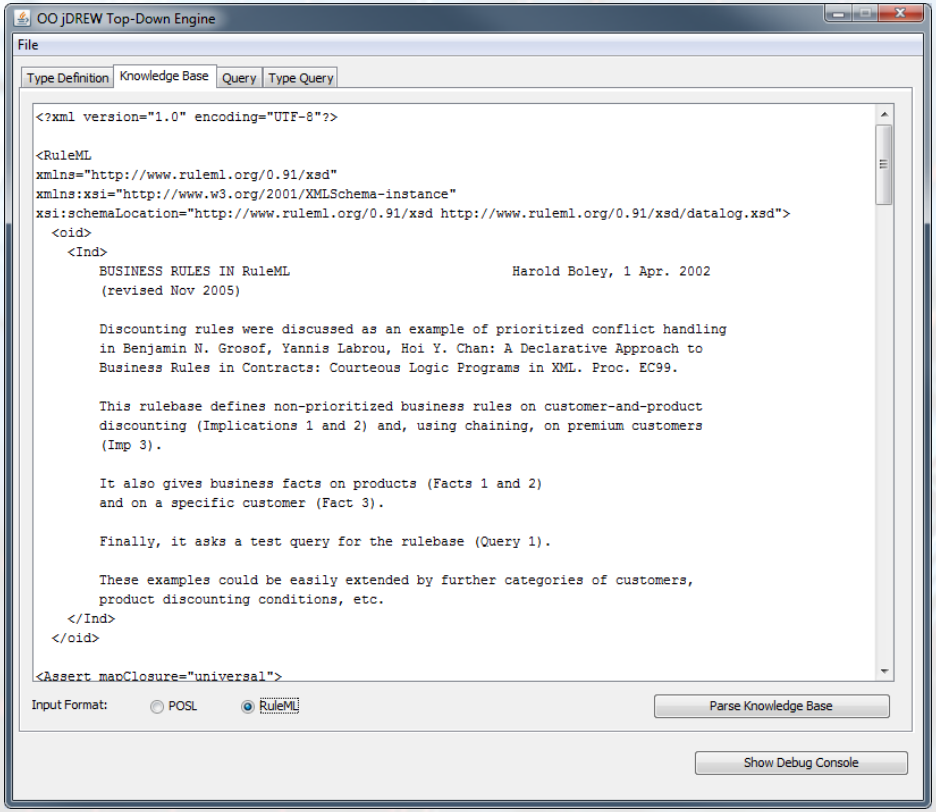


GUI - clipboard access / copy & paste (2)



GUI – font sizes (1)

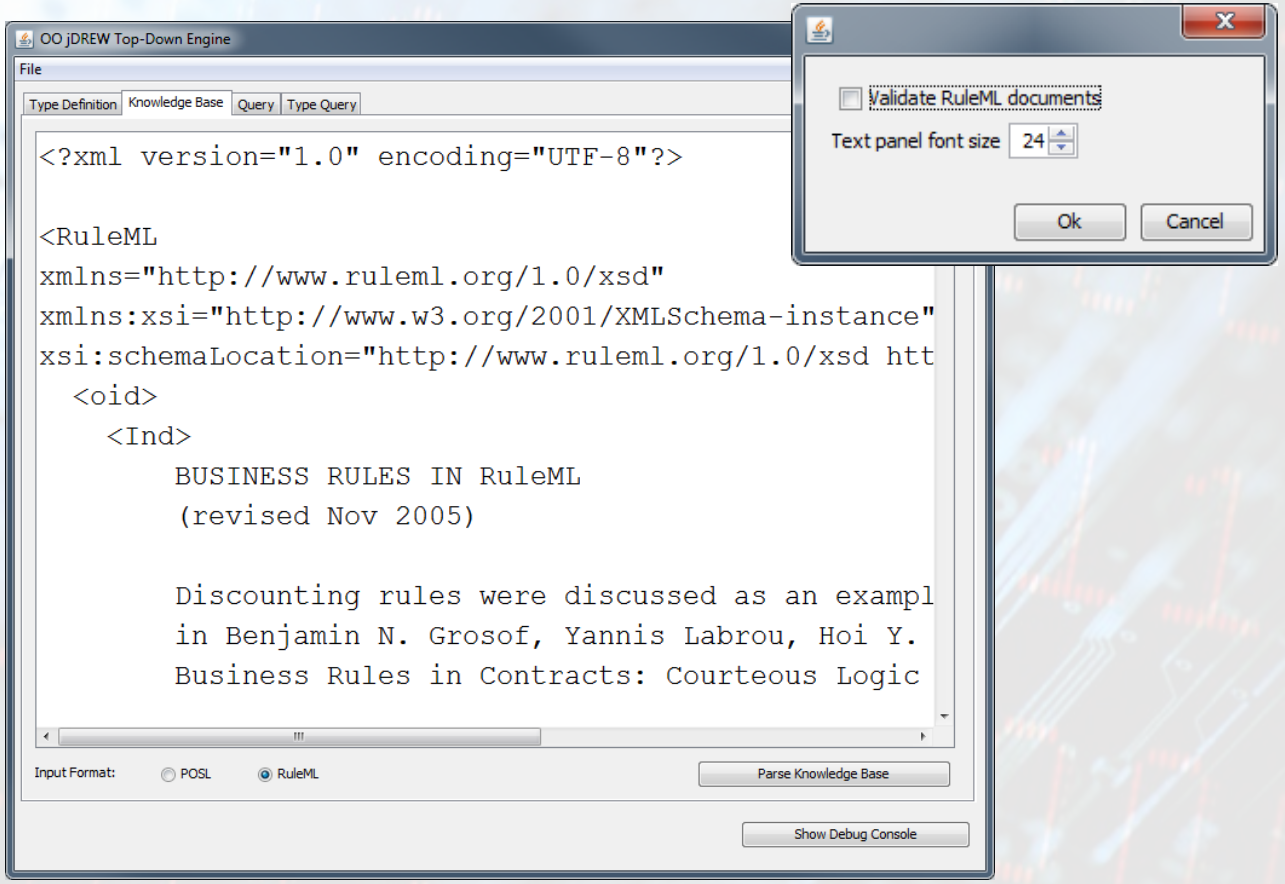
- The GUI components do now apply the "system look and feel" given by the underlying operating system



UImanager

GUI – font sizes (2)

- The text field font sizes are now scalable



Differences: RuleML 0.91 vs. RuleML 1.0

- Role tags
  - **<head> and <body> replaced by <then> and <if>**
  - <lhs> and <rhs> replaced by <left> and <right>
  - New tag for <RuleML> root: <act>
- Attribute and type tag changes

  → Does not affect OO jDREW's internal reasoning engines

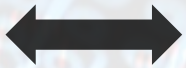
Parser and implementation issues (OO jDREW 0.961)

- KB: Only support for stripe-skipped syntax (e.g. omit <then> <if> or <head> <body>)
- Not accepted by parser:
  - Root element (RuleML)
  - <oid> and further role tags (i.e. <edge>)
- Many undocumented parts in source code
- Duplicate code (~97%)
- Missing support for RuleML 1.0

Digression: “Strip-Skipping”

- Example: striped and strip-skipped syntax

```
<Implies>
   <if>
   if_content
   </if>
   <then>
   then_content
   <then>
</Implies>
```



```
<Implies>
   if_content
   then_content
</Implies>
```

New objectives (RuleML 1.0 parser)

- Understand current parser implementation
- Implementation of non-striped syntax parser
- Merge duplicated parser code into one consistent RuleML parser
- Outsource hard-coded and duplicate variables (e.g. tag names, substring, …)
- Follow the single responsibility principle

Changes and features in RuleML 1.0 parser (1)

- Accepts RuleML role tags where applicable
  (e.g. <act>, <oid>, <left>, <right>, …)
- Support for RuleML 0.88 up to version 1.0
- Stripe-skipped and striped syntax support
  - Including backward compatibility
- Optional XML validation (XSD)
  - Configurable via user interface (UI)

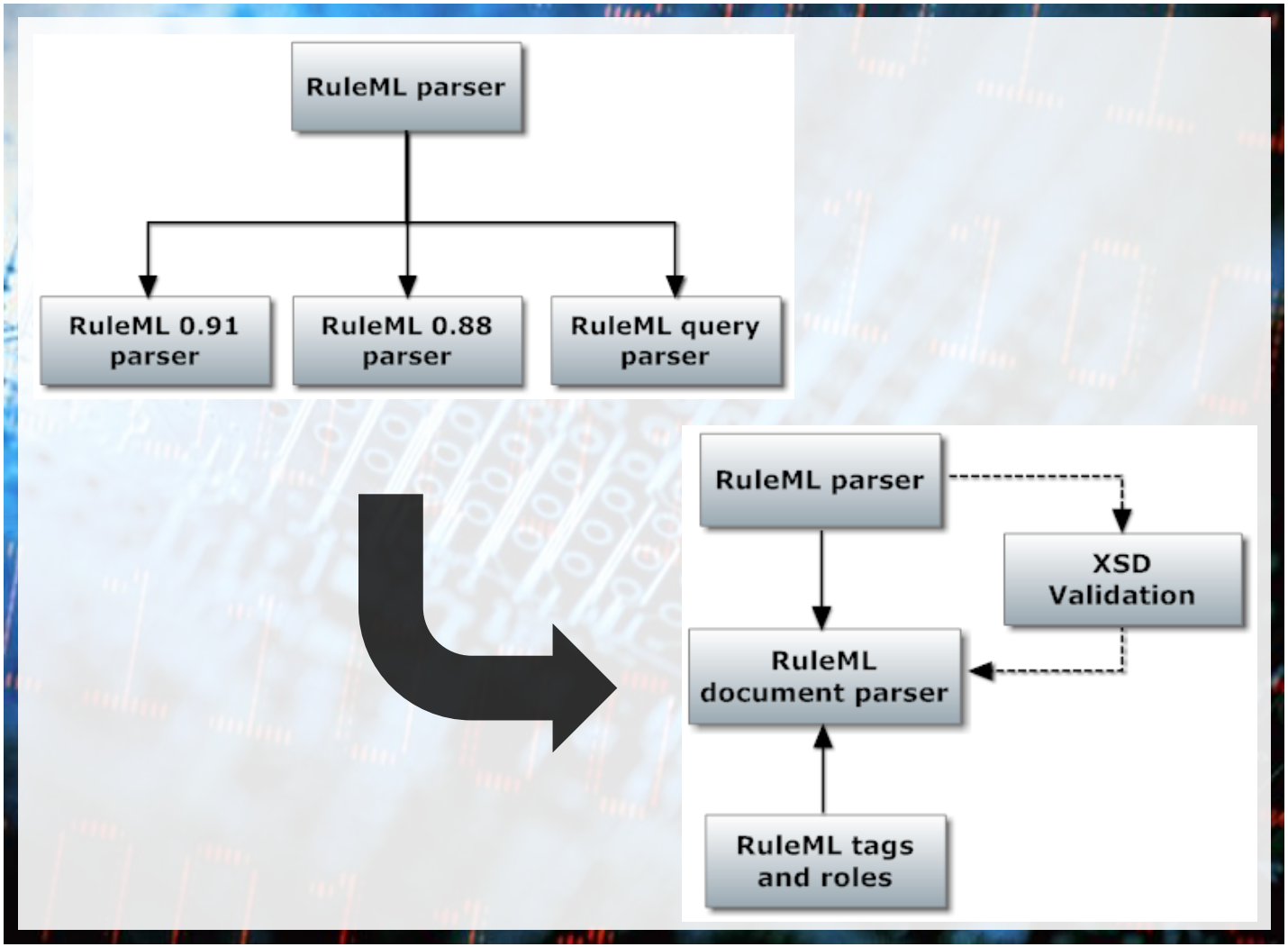
Changes and features in RuleML 1.0 parser (2)


```
<RuleML>
    <Assert mapClosure="universal">
      <formula>
          <Implies>
              <if>
                <And>
                    <formula>
                        <Atom>
                            <op><Rel>buy</Rel></op>
                            <arg index="1"><Var>person</Var></arg>
                            …..
                        </Atom>
                    </formula>
                    <formula>
                        <Atom> …… </Atom>
                    </formula>
                </And>
              </if>
              <then>….. </then>
…
…
```

Source: http://ruleml.org/1.0/

Maintainability and extensibility (1.0)

- Refactoring and code clean-up
  - Tag name class
  - Object instantiation (e.g. Integer)
  - Removed duplicate code
- Single responsibility principle (methods, classes, packages)
- Enhanced JavaDoc (source code documentation)
- Strict typing (future Java support)

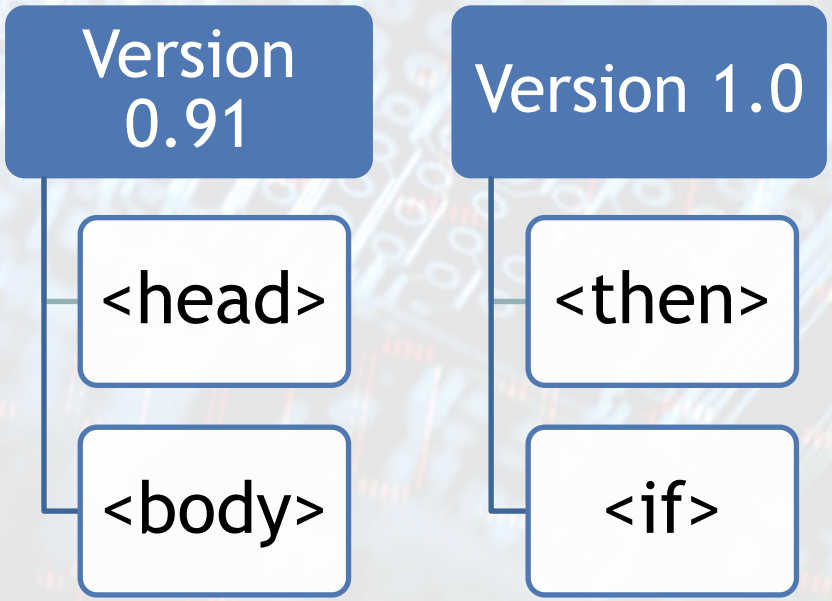


Issues during implementation

- Interdependencies
  - POSL parser
  - Internal data structures
  - Forward/backward reasoning engine (Hornlog)
- Complexity of RuleML parsing
  - Context sensitive role tags (elements have different children at different locations, i.e. <op>)
- Specification
  - No explicit description for attribute mapping

Test cases (1)

- Most of the changes applied to RuleML 1.0 do only affect few XML tags which results in a small set of test cases
- e.g.:



Test cases (2)

- Parser does accept valid RuleML 0.91 and 1.0 syntax (in striped and striped-skipped syntax) instead of only accepting stripe-skipped syntax
- The previous parser outputs an error when parsing a valid RuleML 0.91 document with striped syntax

Error message

Second element of Implies should always be an Atom or Neg element

Simple Test Case with two rules (1)

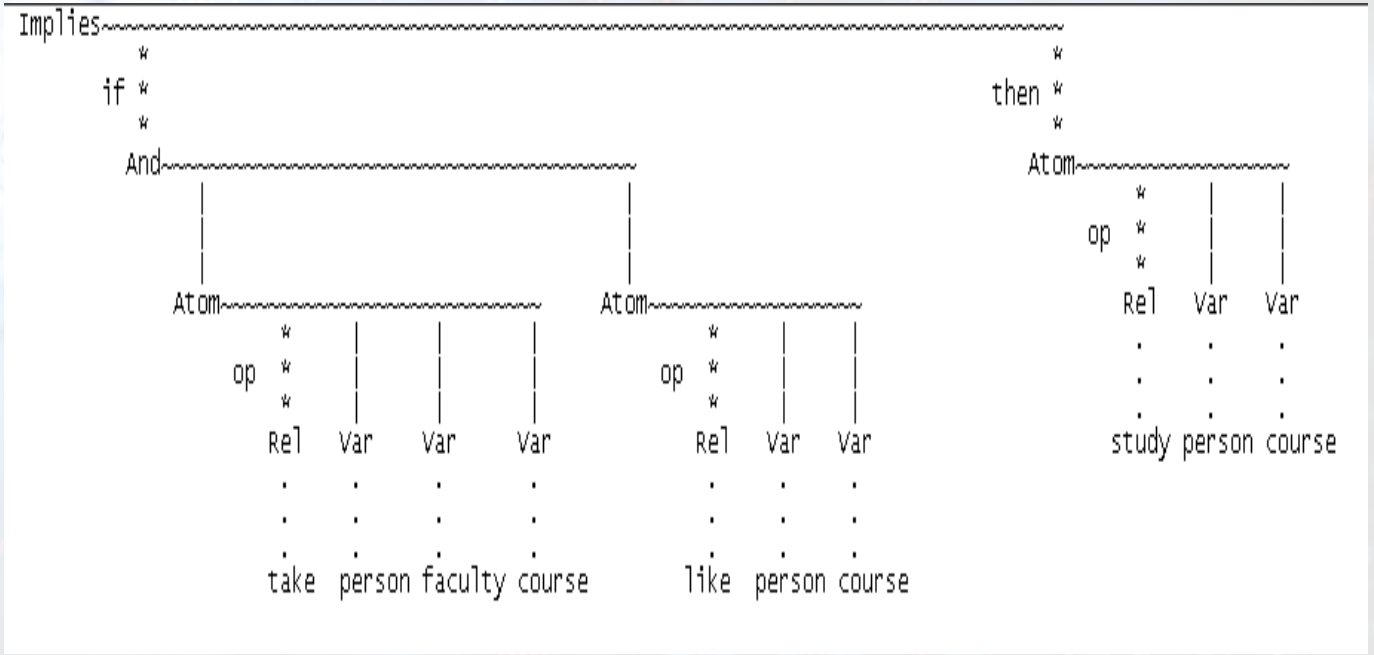
- In English

The first rule is: a person studies a course if a person takes a course in a faculty and a person likes a course

The second rule is: a person takes a course in a faculty if a faculty has a course for a person

Simple Test Case with two rules (2)

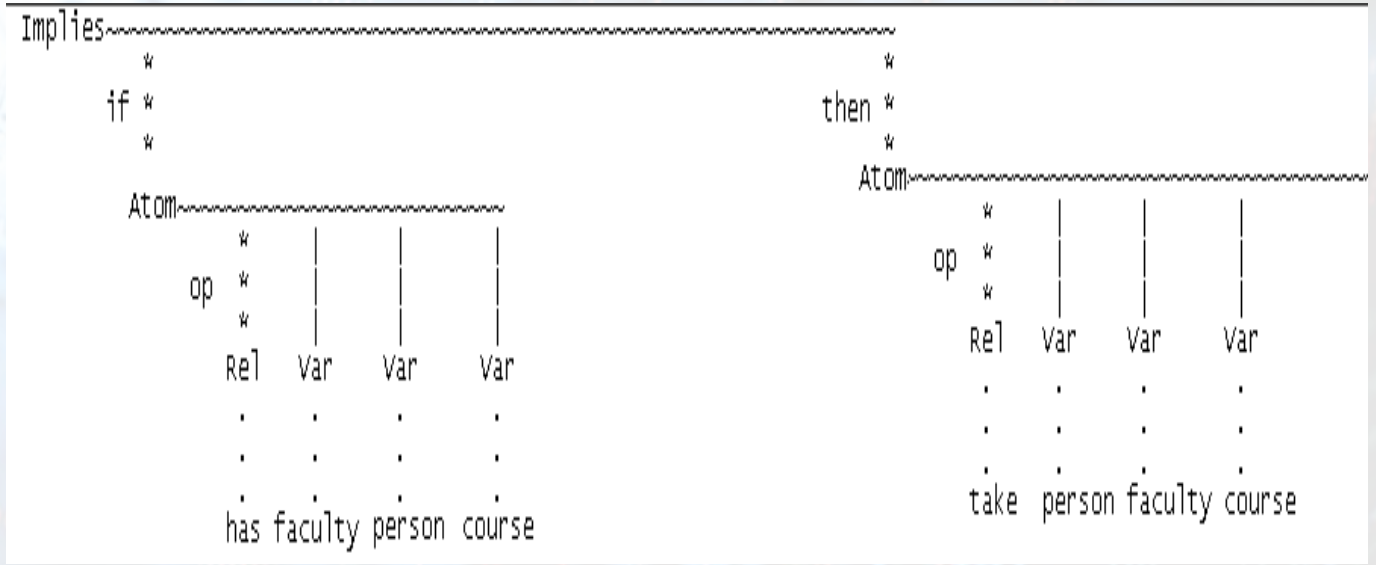
- As an OrdLab Tree: rule1



Source: based on http://ruleml.org/1.0/exa/Datalog/own.ruleml

Simple Test Case with two rules (3)

- As an OrdLab Tree: rule2



Source: based on http://ruleml.org/1.0/exa/Datalog/own.ruleml

DEMO

Limitations of the new parser

- Relies on XSD for validation
  - The official XDSs don't agree with the RuleML standard
  - Validation requires network connectivity
  - Doesn't differentiate between RuleML dialects
  - Skips many tags
  - May not take every feature of the reasoning engine into account

Conclusion and outlook

- New parser covers the most common RuleML use cases
- Project ready to accept third-party contributions
- Integrate the other project team's changes
- More Refactoring
  - e.g. make the RuleML parser independent of the POSL parser
  - More test cases
  - Make reasoner understandable

Thank you for your attention!

Questions?

omar.alsaiari@gmail.com, k.almutairi@hotmail.com,
{nass11.s, c.fabbricatore, m.zucker}@unb.ca