Master Project Proposal

Paul Spencer: 11721677
University of Amsterdam
paul.spencer@student.uva.nl

January 2021

Project details

- **Project title:** Projecting Drools
- **Host organization:** Khonraad Software Engineering B.V.
- **Host supervisor:** Toine Khonraad, MD, a.khonraad@khonraad.nl +31 (0)6 51 61 47 55

1 Project summary

Drools[1] is an open-source production rule system for complex event processing, using an implementation of the Rete algorithm[5]. It has its own Domain Specific Language (DSL), in which rules are described. These are stored in Drools (.drl) files. When there are many rules in a collection of Drools files it can be difficult to reason about them and how they interact.

Although Drools is nearly 20 years old and has wide use, it does not have strong IDE support. This masters project will approach fixing this deficit by creating a projectional editor available in IntelliJ, currently the most used IDE[9].

We will recreate this DSL using a language workbench capable of creating projectional editors. On top of this newly modelled DSL, we will create new and different projections of the code. Editing in language workbenches comes in two forms, either free-form text editing or, less frequently, projectional editing[4]. Projectional editing is a method of bypassing the parser and programming directly into projections of the Abstract Syntax Tree.

In this project we will use the open-source language workbench Meta Programming System (MPS) from JetBrains[8]. MPS is built around the projectional editing paradigm. There is no existing implementation of the Drools language in MPS.

Khonraad's product is a SaaS service that serves municipalities in the field of mental health and domestic violence. It goes without saying that its data must not be accessed by the wrong people. Drools is the technology choice Khonraad

made to govern this critical function of data access. Thus, they would like to improve the reasonability of this code.

This project will attempt to answer the following research questions:

- **RQ1:** What is the value of using a projectional editor?
- **RQ2:** How can the benefits of projectional editing be applied to Drools?
- **RQ3:** Which refactorings and projections will most benefit the reasonability of Drools rules?

2 Problem analysis

The mental health care and coercion laws in the Netherlands, Wvggz, Wzd, and Wth, provides agencies the ability to intervene in domestic violence, psychiatric disorders, and illnesses. Khonraad's system facilitates reporting and communication between municipalities, police, judiciary, lawyers, mental health care, and countless social care institutions. The system has 15,000 users and is available 24/7. Configuration and administration use complex matrices of compliance mechanisms, access user rights and communication settings. The sensitivity of the personal data, being medical and criminal, means security is of utmost importance. The security against data loss, preventing unlawful disclosure and guaranteeing availability, especially during a crisis situations, are crucial. Demonstration of the correctness of the, often changing, configuration is a major concern in the company.

In the current situation, configuration is done in a business rule system. This is Drools, a DSL from JBoss, a subsidiary of RedHat. Drools is a framework for Rule-Based development. The DSL is a textual representation of the abstractions of the rules and must be compiled to see if it is valid and works. Editing programs in a text editor means that you must match the syntax for the parsers to transform the text into an AST.

Projectional editors are editors in which a user edits the abstract syntax tree directly without using a parser[24]. This potentially allows for an almost unlimited language composition and flexible notations. Similar to the MVC Pattern, changes in one projection of the AST will instantly be visible and editable in another projection[7].

The problem of a lack of useful visualization for Drools has been known as far back as 2011. Kaczor, et al[10] proposed a method of visualising Drools. There have also been a few commercial tools to help. However, these all suffer from the parsing issue and lack of immediate feedback. We are of the opinion that our approach will lend itself to a superior experience.

3 Research method

RQ1 will be answered with a mapping of the field of projectional editing. This will follow the prescriptions of Kitchenham et al.[11]. Gregor[6], gives "A Tax-

onomy of Theory Types in Information Systems Research". To answer RQ2 and RQ3, we will conduct what she calls "Type V: Theory for Design and Action". The criteria for success of this type of research "include utility to a community of users, the novelty of the artefact, and the persuasiveness of claims that it is effective".

We have observed the difficulty that developers have trying to reason about and edit large collections of Drools files. We hypothesize that developers can be presented with different views on their code that will allow them to better understand the code. The problem we wish to solve - how to improve the ability to reason about large collections of drools rules - lends itself to the technique of projectional editing. Thus, we will apply projectional editing techniques, through the MPS language workbench to the Drools language. The novelty of our approach will be to create new view types specific to the needs of a Drools programmer.

We will be relying on MPS as well as other open-source components to work together with acceptable performance such that the user experience is acceptable. The reason we chose MPS is that it is the most developed of the free and open source projectional editing language workbenches, found in a study of the state of the art in Language workbenches[4].

Figure 1 shows the JetBrains IDE for MPS, which is also free and open-source. In this you create the Concepts, Generators, Type Systems, etc., that are necessary to define the language. You also create editors, which gives your language out of the box IDE support.

Our designs of the projections, which will run in parallel to the Drools language modelling, will depend in part on the outcome of research carried out in the first period. Whether our design is appropriate with regards to performance and functionality is a risk. Whether we can achieve usefulness in our projections also presents a risk. This is where I expect to gain the biggest benefit of literature review and academic supervision.

The prototype will consist of a base of the Drools language, re-defined in MPS. On top of this will be a set of different projections of the AST. Whilst we have not decided on the projections yet, some examples may include:

- Visualization of order of execution.
- Spreadsheet-like decision tables.
- "Group-by" fact, query or function usage.

The major tasks in this prototype development will be:

- Modelling the Drools language.
- Developing the alternative projections.

The prototype itself will be validated by working and being novel. However, if time permits, the hypothesis of the usefulness of the projections will be validated through developer use surveys.

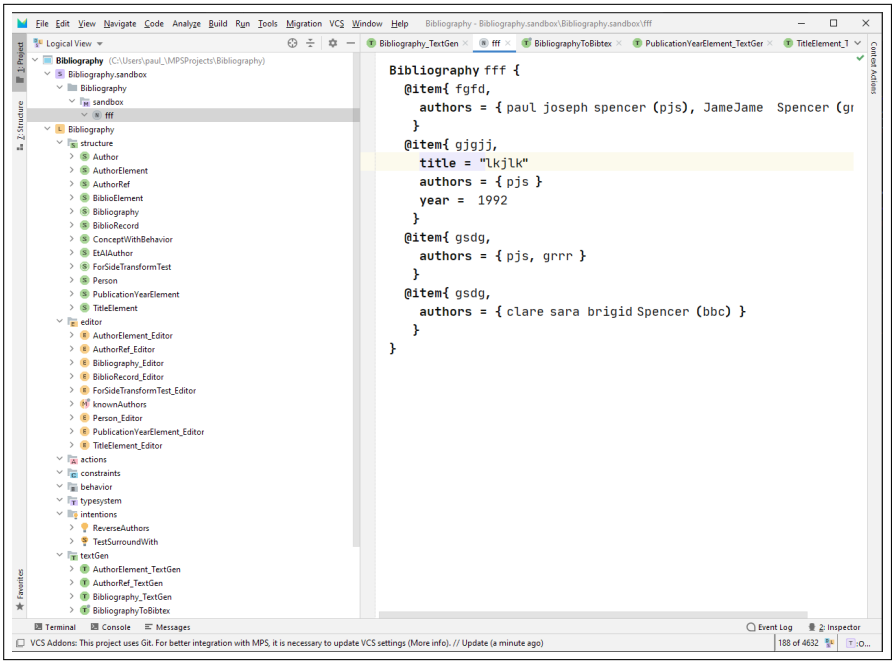

Figure 1: MPS IDE

4 Expected results of the project.

We expect the following from this project:

- We will be able to model Drools in MPS.
- A suite of novel and useful projectional editors for the Drools language.
- We will reduce the thought to execution cycle for Drools Developers. resulting in a reduction of the "cognitive distance and representation impedance mismatch"[22] that they currently experience.

A happy side effects of this project is that the following open-source products will become generally available:

- An improved Drools Editor plugin for those using the JetBrains IntelliJ community edition.
- An MPS implementation of Drools that can be used for cross-generation by other MPS Language implementations.

5 Required expertise for this project.

Table 1 shows our expected and actual expertise levels in the technologies and practices required to complete this project.

Skill	Required	Acquired	Notes
MPS	★★★★★	★★½☆☆	Currently taking various courses.
Drools	★★★½☆	★★☆☆☆	The language is simple.
Java	★★★★☆	★★★☆☆	15 years of C#, these are similar.
Swing	★★★☆☆	☆☆☆☆☆	never played with this.
Language Design	★★★☆☆	★★☆☆☆	more for deconstructing Drools. than creating a new language.

Table 1: Expertise required.

6 Timeline

This project consists of two main parts. First is modelling the Drools structure, behaviour, constraints, editors, and generators. The second will be creating non-standard projections of the structure.

Time will be allocated as 20 hours of my work time per week will be dedicated to design and development of the software. Currently estimating 4-8 hours at the weekends to research and project writing. There is an added period of 4 weeks at the end to rewriting the thesis. This is shown in the Gantt Chart in figure 2

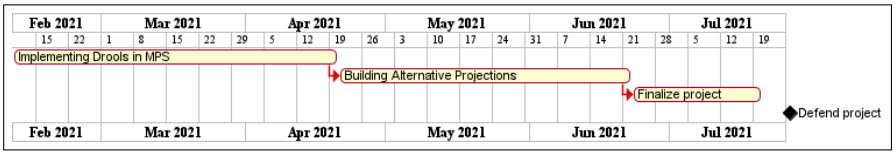

Figure 2: Predicted timeline.

7 Risks

Table 2 shows the main risks we see to this project.

Description	Risk Level	Contingency
Project goals are too ambitious	★★★☆☆	Reduce Drools implementation to a useful subset and reduce the number of projections. Jettison the mapping study.
MPS is not as flexible as needed	★★☆☆☆	We are very sad, but the papers about mbeddr indicate low risk.
I'm a terrible MPS developer	★★★★★	Currently taking training and reading lots of books.
Academic overlap	★★☆☆☆	Panic or ignore.

Table 2: Project Risk.

8 Literature survey

To get an overview of the field we mainly looked at MPS and Drools based papers. For MPS we started with an expert recommendation and did some forward and backward snowballing. A Google Scholar search produced one Drools papers with work on code visualization. The MPS papers and associated DSL papers covered some aspects of visual projectional editing, especially the papers relating to the produce mbeddr.

Table 3 summarizes the papers and books investigates in preparation for this project.

During our research we will keep a document database, using the Zotero Personal research assistant software[20]. Also we will create an annotated bibliography of the most relevant papers.

citations	Papers				
	Creating DSLs	How Drools works	Comparing Workbenches	How MPS works	Projectional Editing
[2]		⊕			
[3]			⊕	⊕	
[4]			⊕		
[7]					⊕
[10]		⊕			
[12]		⊕			
[13]			⊕	⊕	
[14]				⊕	
[15]				⊕	
[16]				⊕	
[17]		⊕			
[18]				⊕	
[19]	⊕			⊕	
[21]		⊕			
[23]				⊕	
[24]	⊕		⊕	⊕	
[25]	⊕				
[26]				⊕	⊕
[27]				⊕	
[28]					⊕
[29]				⊕	⊕
[30]	⊕		⊕	⊕	⊕
[31]				⊕	
[32]	⊕			⊕	

Table 3: Papers about the Drools, MPS and Language workbenches

References

[1] Paul Browne. *JBoss Drools business rules*. Packt Publishing Ltd, 2009.

[2] Erwin De Ley and Dirk Jacobs. "Rules-based analysis with JBoss Drools: adding intelligence to automation". In: *Proceedings of ICALEPCS 2011* (2011), pp. 790–793.

[3] Sebastian Erdweg et al. “Evaluating and comparing language workbenches: Existing results and benchmarks for the future”. In: *Computer Languages, Systems & Structures* 44 (2015), pp. 24–47.

[4] Sebastian Erdweg et al. “The state of the art in language workbenches”. In: *International Conference on Software Language Engineering*. Springer. 2013, pp. 197–217.

[5] Charles L Forgy. “Rete: A fast algorithm for the many pattern/many object pattern match problem”. In: *Readings in Artificial Intelligence and Databases*. Elsevier, 1989, pp. 547–559.

[6] Shirley Gregor. “The nature of theory in information systems”. In: *MIS quarterly* (2006), pp. 611–642.

[7] Stian M Guttormsen, Andreas Prinz, and Terje Gjøsæter. “Consistent Projectional Text Editors.” In: *MODELSWARD*. 2017, pp. 515–522.

[8] *Jetbrains MPS Product Page*. `https://www.jetbrains.com/mps/`. Accessed: 2021-01-19.

[9] *JRebel Java Technology Report*. `https://www.jrebel.com/blog/2020-java-technology-report`. Accessed: 2021-01-27.

[10] Krzysztof Kaczor et al. “Visual design of Drools rule bases using the XTT2 method”. In: *Semantic methods for knowledge management and communication*. Springer, 2011, pp. 57–66.

[11] Barbara Ann Kitchenham, David Budgen, and Pearl Brereton. *Evidence-based software engineering and systematic reviews*. Vol. 4. CRC press, 2015.

[12] Narendra Kumar, Dipti D Patil, and Vijay M Wadhai. “Rule based programming with Drools”. In: *International Journal of Computer Science and Information Technologies* 2.3 (2011), pp. 1121–1126.

[13] Sofia Meacham, Vaclav Pech, and Detlef Nauck. “Classification Algorithms Framework (CAF) to Enable Intelligent Systems Using JetBrains MPS Domain-Specific Languages Environment”. In: *IEEE Access* 8 (2020), pp. 14832–14840.

[14] Domenik Pavletic et al. “Extensible debuggers for extensible languages”. In: *GI/ACM WS on Software Reengineering* (2013).

[15] Vaclav Pech, Alex Shatalin, and Markus Voelter. “JetBrains MPS as a tool for extending Java”. In: *Proceedings of the 2013 International Conference on Principles and Practices of Programming on the Java Platform: Virtual Machines, Languages, and Tools*. 2013, pp. 165–168.

[16] Andreas Prinz. “Multi-level Language Descriptions.” In: *MULTI MoDELS*. 2016, pp. 56–65.

[17] Mark Proctor. “Drools: a rule engine for complex event processing”. In: *International Symposium on Applications of Graph Transformations with Industrial Relevance*. Springer. 2011, pp. 2–2.

[18] Daniel Ratiu, Vaclav Pech, and Kolja Dummann. "Experiences with teaching MPS in industry: towards bringing domain specific languages closer to practitioners". In: *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE. 2017, pp. 83–92.

[19] Daniel Ratiu, Markus Voelter, and Domenik Pavletic. "Automated testing of DSL implementations—experiences from building mbeddr". In: *Software Quality Journal* 26.4 (2018), pp. 1483–1518.

[20] *Roy Rosenzweig Center For History And New Media. Zotero product page.* `https://www.zotero.org/`. Accessed: 2021-01-24.

[21] Kay-Uwe Schmidt, Roland Stühmer, and Ljiljana Stojanovic. "Blending complex event processing with the rete algorithm". In: *Proceedings of iCEP2008: 1st International Workshop on Complex Event Processing for the Future Internet*. Vol. 412. Citeseer. 2008.

[22] Tijs van der Storm and Felienne Hermans. "Live literals". In: *Workshop on Live Programming (LIVE)*. Vol. 2016. 2016.

[23] Markus Voelter. "Fusing modeling and programming into language-oriented programming". In: *International Symposium on Leveraging Applications of Formal Methods*. Springer. 2018, pp. 309–339.

[24] Markus Voelter. *Generic tools, specific languages*. Citeseer, 2014.

[25] Markus Voelter et al. *DSL engineering: Designing, implementing and using domain-specific languages*. dslbook. org, 2013.

[26] Markus Voelter et al. "Lessons learned from developing mbeddr: a case study in language engineering with MPS". In: *Software & Systems Modeling* 18.1 (2019), pp. 585–630.

[27] Markus Voelter et al. "Shadow models: incremental transformations for MPS". In: *Proceedings of the 12th ACM SIGPLAN International Conference on Software Language Engineering*. 2019, pp. 61–65.

[28] Markus Voelter et al. "Towards user-friendly projectional editors". In: *International Conference on Software Language Engineering*. Springer. 2014, pp. 41–61.

[29] Markus Voelter et al. "Using C language extensions for developing embedded software: a case study". In: *Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*. 2015, pp. 655–674.

[30] Markus Voelter et al. "Using language workbenches and domain-specific languages for safety-critical software development". In: *Software & Systems Modeling* 18.4 (2019), pp. 2507–2530.

[31] Premysl Vysoky. "Grammar to JetBrains MPS Convertor". In: (2016).

[32] Andreas Wortmann and Martin Beet. "Domain specific languages for efficient satellite control software development". In: *ESASP* 736 (2016), p. 2.