UNIVERSITEIT VAN AMSTERDAM

MASTERS PROJECT

Representation Mismatch Reduction for Development in Rules-Based Business Engines

Author:
Paul SPENCER

Supervisor:
Dr. Clemens GRELCK

A thesis submitted in fulfillment of the requirements for the degree of Master of Software Engineering

in the

Graduate School of Informatics
Faculty of Science

September 13, 2021

UNIVERSITY OF AMSTERDAM

Declaration of Authorship

I, Paul SPENCER, declare that this thesis titled, "Representation Mismatch Reduction for Development in Rules-Based Business Engines" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. Except for such quotations, this thesis is entirely my work.
- I have acknowledged all of the main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ______________________________

Date: ______________________________

UNIVERSITEIT VAN AMSTERDAM

Abstract

Graduate School of Informatics

Faculty of Science

Master of Software Engineering

Representation Mismatch Reduction for Development in Rules-Based Business Engines

by Paul SPENCER

Context: Declarative rules engine languages, such as Drools, can become difficult to reason about when there are many rules.

Objective: This project investigates how different projections of the code can ease the comprehensibility of the code.

Method: We created an implementation of the Drools language using the MPS language workbench and made innovative projections of large ASTs.

Results:

Keywords: projectional editing; Rules Engines; MPS; Drools

Paper type: Research paper

Acknowledgements

We would like to acknowledge the School of The Gradate school of Informatics in the Faculty of Science at The University of Amsterdam for their guidance, specifically Dr. Clemens Grelck, who has been a supportive, understanding, and available academic adviser.

We received inspiration from the Strumenta Languages engineering community. Specifically we would like to thank Federico Tomasetti, who shared with me his model of a rules engine in MPS.

Also we would like to thank Václav Pech from JetBrains for the course he created and the time he spent with me explaining MPS. Further, Sergej Koščejev from JetBrains helped us with specific MPS issues during his Office hours.

Other prolific output that terrifically helped our research and development was the Heavy Meta YouTube series from Kolja Dummann and the dozens of papers and books from Markus Voelter, both currently working at Itemis A.G.

Our greatest thanks go out to Toine Khonraad, an alum of this course, who provided me with moral and monetary support, as well as wisdom and friendship that aided in the completion of this, my fourth attempt at getting this project behind me. Without his constant mantra of simplify, simplify, simplify, we would still be implementing the Drools languages now without having made a single projection.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The limits of my language mean the limits of my world.

Logico-Tractatus Philosophicus
Ludwig Wittgenstein

1.1 Problem Statement

Drools is a language that shares an unfortunate characteristic with many other rules languages. Namely, It is verbose and can contain many rules that can interact without an apparent visual connection. As Forgy[1] points out, for rules languages in general, "[they] have another property that makes them particularly attractive for constructing large programs: they do not require the developer to specify in minute detail exactly how the various parts of the program will interact". This property leads to very large and difficult to reason about collections of implicitly connected rules.

Reasoning over a small number of rules is already surprisingly hard. Our host organization has many rules and, thus, reasoning about them is particularly challenging.

One approach to tackle comprehensibility could be to consider Miller's Law[2]. This states that an average human can hold in his short-term memory 5-9 objects, which is often an argument for succinct code. The argument is that the developer must store anything that is not immediately in her vision in her memory. With it being impractical to reason about code that she cannot recall, the fewer relevant items to her reasoning that are out of view, the easier it is to reason about the code.

We have observed the difficulty developers have to try to reason about and edit collections of Drools files. We hypothesize that when we present developers with different views on their code, they can better understand it. The problem we wish to solve - how to improve the ability to reason about sizeable collections of Drools rules - we believe lends itself to the technique of projectional editing. By using projections to improve feedback whilst coding, we believe that this can reduce the representation impedance mismatch that hampers the developer's reasoning.

Our intention is not to override the language engineers who have spent many years developing this language and its ecosystem. In contrast, it is to augment the current developer experience. Thus, the problem considered in this thesis is how to present large sets of rules to a developer to interact with them easily.

1.2 Research Questions

To reason about a large code base of rules engine code effectively, a different presentation is needed. This presentation should allow a more precise organization whilst remaining interactive. We can formulate the following research questions based on the discussion in the preceding sections, being:

- **Main research question:** "How can projectional editors and DSLs be combined to address feedback mechanisms for developers in the context of reasoning about rules in a rule-based business engine?"

This question requires knowing if it is possible with current tooling. Thus we would like to answer the question:

- **RQ 1:** "What is the current state of language workbenches supporting projectional editing?"

Finally, we specifically would like to know how to improve the ability to reason about the business rules engine, so we ask the question:

- **RQ 2:** "Which projections can help developers get appropriate feedback about rules?"

1.3 Contributions

This thesis proposes a representation of business rules in a concise and readable format that could solve comprehensibility issues resulting from large codebases of business rules. The implementation behind the approach relies on language engineering and projectional editing. We developed a stand-alone opensource solution on a limited implementation of Drools. The underlying Drools implementation can be used as a base language for model-to-model generation by the MPS ecosystem.

1.4 Project Context

Khonraad Software Engineering, a subsidiary of Visma, hosted this investigation. Khonraad provides mission-critical services focussed on the automation of workflows at the cross-section of local government and healthcare. Specifically, Khonraad facilitates the mental health care and coercion laws in the Netherlands - WVGGZ, WZD, and WTH - which allow agencies 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 many social care institutions. The system has 15,000 users and is available 24/7.

Configuration and administration require complex matrices of compliance mechanisms, access user rights and communication settings. The sensitivity of the personal data, being both medical and criminal, means security is of utmost importance. The security against data loss, preventing unlawful disclosure and guaranteeing availability, especially during crises, is crucial. Demonstration of the correctness of the configuration is a significant concern in the company.

This work environment allows us to work on an existing project, where success will impact the lives of those in critical need. Khonraad has its implementations in the Drools language that have evolved together with the iterations of the laws. Over the years, the evolution of the codebase means that we deal with real-life issues and not just thought experiments.

1.5 Thesis Outline

We start in chapter 2 with the required background information on rules engines and projectional editing. In chapter 3, we present our strategies for answering the research questions. Chapter 4 presents the results of our various approaches. In chapter 5, we discuss both the validity of the work and the implications of the findings. We discuss the implications of this study in chapter 6. Finally, we present the conclusions in chapter 7.

Chapter 2

Background

This chapter gives the background information required on rules engines and projectional editing. It presents the specific case of the rules engine that we will be using for our investigation: Drools. Next, we delve into the world of projectional editing. Finally, it presents the specific projectional editing tool we will be using: JetBrains MPS.

2.1 RulesEngines

2.1.1 What is a Rules Engine?

In this section, we will describe what a rules engine is and a little of its history.

The Aristotelian doctrine of essentialism declares that a thing has essential properties and properties that are accidental. If one takes away accidental properties, then the thing remains the thing. If one takes away essential properties, the thing is no longer the thing. If the thing is a business application, then its essential properties are its business rules.

Simply put, business rules are the principles or regulations by which an organization carries out the tasks needed to achieve its goals. When adequately defined, it is possible to encode these rules into statements that define or constrain some business organizational behaviour. A rule consists of a condition and an action. When the condition is satisfied, then the action is performed. More formally, business rules are the implication in the logical principle of Modus Ponens.

When described like this, one could imagine this as just the “if-then” logic frequently used in traditional programming. One would not be wrong. However, in traditional programming, representing all the combinatorial outcomes can become complex. In the typical application architecture, developers tend to distribute rules throughout the source code or database. Each additional rule leads to more fragility.

We find descriptions of these rules in the design documentation or user manuals. However, as applications evolve, documentation gets out of sync with the codebase. Once desynchronization occurs, to know the rules governing the application, one has to navigate the codebase and decode the rules from often scattered locations.

A rules engine is also known as a Business Rules Engine, a Business Rules Management System or a Production Rules System. The goal of a rules engine is to abstract business rules into encoded and packaged logic that defines the tasks of an organization with the accompanying tools that evaluate and execute these rules. Simply put, they are where we evaluate our rules. Rules engines match rules against facts and infer conclusions. Returning to the Modus Ponens comparison:

$$\begin{array}{l} \quad p \\ \quad \underline{p \rightarrow q} \\ \therefore q \end{array}$$

If the premise p holds and the implication $p \rightarrow q$ holds then the conclusion q holds. In terms of a rule engine and business rules this could be seen as:

1. the rules engine gathers the data for the premise: p
2. it examines the business rules as the implications: $p \rightarrow q$
3. it executes the conclusion: q

Rules engines follow the recognize-act cycle. First, the match, i.e. are there any rules with a true condition? Next, they carry out conflict resolution, pick the most relevant matching rules. They then perform the actions described in the rule. Then back to the matching step. If they make no more matches, they terminate the cycle.

Rules Engines are declarative, focussing on the what of the rules, not the how of the execution. Date[3] describes a rules engine role as "to specify business process declaratively, via business rules and get the system to compile those rules in to the necessary procedural (and executable) code." Fowler[4] describes a rules engine as follows: " ... providing an alternative computational model. Instead of the usual imperative model, which consists of commands in sequence with conditionals and loops, a rules engine is based on a Production Rule System. This is a set of production rules, each of which has a condition and an action ...".

Some of the advantages of using a rules engine include:

- The separation of knowledge from its implementation logic.
- The externalization of business logic.
- Rules can be human-readable.

In summary, a rules engine is the executor of a rules-based program, consisting of discreet declarative rules which model a part of the business domain.

Rule engines arose from the expert systems of the late 70s and early 80s. Expert systems initially had three primary techniques for knowledge representation: Rules, frames and logic[5]. "The granddaddy" of the expert systems, MYCIN, relied heavily on rules-based knowledge representation[6] rather than long inference chains. MYCIN was used to identify bacteria and recommend antibiotic prescriptions. MYCIN and its progenitor, DENDRAL, spawned a whole family of Clinical Decision Support Systems that pushed the rules engine technology until the early 1980s. Research into rules engines died out in the 1980s as it fell out of fashion.

Early in their existence, the rules engines hit a limiting factor because the matching algorithms they used suffered from the utility problem, i.e. the match cost increased linearly with the number of examined rules. Charles Forgy's efficient pattern matching Rete

Product		Developer	licence type
CLIPS	[7]	NASA	open source
Drools	[8]	JBoss/RedHat	open source
BizTalk Business Rule Engine	[9]	Microsoft	proprietary
WebSphere ILOG JRules	[10]	IBM	proprietary
OpenRules	[11]	OpenRules	open source

TABLE 2.1: Rules engine products

algorithm[1] and its successors solved this problem. This algorithm works by modelling the rules as a network of nodes where each node type works as a filter. A fact flows through the filters of this network. The pre-calculation of this network is what provides the performance characteristics.

The first popular rules engine was Office Production System (OPS) from 1976. In 1981 OPS5 added the Rete algorithm. Currently, there are a few rules engines in use. We show some of the more commonly used ones in table 2.1.

2.1.2 What is Drools?

JBoss Rules, or more commonly known, Drools, is the leading opensource rules engine written in Java. In this paper, when we use the name "Drools" we are referring to the "Drools Expert" which is the rule engine module of the Drools Suite. Drools started in 2001 but rose to prominence with its 2005 2.0 release. It is an advanced inference engine using an enhanced version of the Rete algorithm, called ReteOO[12], adapted to an object-oriented interface specifically for Java. Designed to accept pluggable language implementations, it can also work with Python and .Net. It is considered one of the most developed and supported rules platforms.

To execute rules, Drools has four major components, as demonstrated in figure 2.1. The production memory contains the rules, and this will not change during an analysis session. The rules are the focus of this thesis, and therefore, we will delve into much more detail later on these.

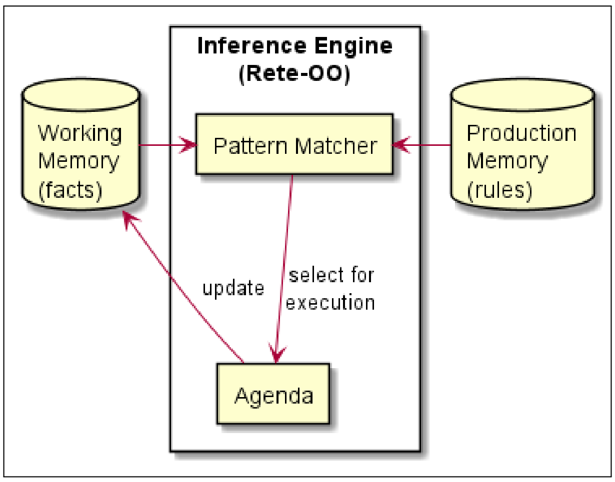


FIGURE 2.1: Drools components.

In Forgy's[1] overview of a rete algorithm, the following steps occur.

1. Match: Evaluate the LHSs of the productions to determine which are satisfied given the current contents of working memory.

2. Conflict resolution: Select one production with a satisfied LHS; halt the interpreter if no productions have satisfied LHSs.

3. Act: Perform the actions in the RHS of the selected production.

4. Re-evaluate: Go To 1.

Figure 2.2 show more detail of how these components interact within Drools to infer a conclusion. First, Drools asserts facts in the working memory. The working memory contains the current state of the facts, which triggers the inference engine. Using the aforementioned Rete-OO algorithm, the pattern matcher will examine the working memory and a representation of the rules from the production memory to determine which rules are true. Drools will then put the rules that match on the agenda. It can be the case that many rules are concurrently true for the same fact assertion. These rules conflict. A conflict resolution strategy will decide which rule will fire in which order from the agenda. The first rule on the agenda will fire. If the rule modifies, retracts or asserts a fact, then the inference loop begins again. We have inferred our conclusion if a rule specifies to halt or there are no matching rules left on the agenda.

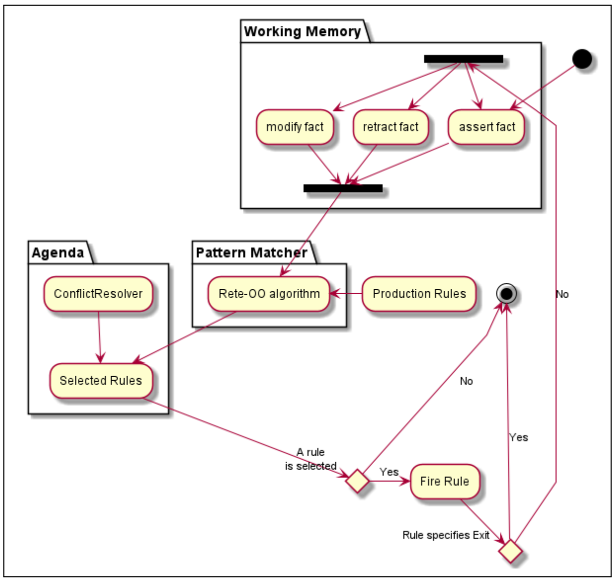


FIGURE 2.2: Drools inference loop

The component we will be focussing on in this paper is the rules. A rules file containing the rules is a text file, typically with a .drl extension. As the rules do not change during execution During execution, the rules do not change, and Drools stores them in production memory.

We do not need to examine the rule file components package, import, global, declare, function and query. We will examine the anatomy of a rule.

A rule consists of 3 parts: attributes, conditions, and consequences. Attributes are optional hints to the inference engine as to how to examine a rule. The conditional, "when", or left-hand side (LHS) of the rule statement is a block of conditions that have to, in aggregate, return true for the asserted fact. If true, then the rule is placed on the agenda. The actions, consequences, “then”, or right-hand side (RHS) of the rule statement contains actions to be executed when the rule is selected.

The LHS is a predicate statement made up of some patterns. The patterns evaluate facts from the working memory. The pattern can match against the existence of facts or facts with matching property conditions. Connectives, such as not, and, and or can combine patterns. The patterns apply to individual facts rather than the group, thus can be seen as first-order predicates.

Variables can be bound to facts that match these patterns for use later in the LHS or for updating the working memory on the RHS.

Drools offers more options for the LHS. We have limited the scope of this paper to the features described thus far.

The RHS can contain arbitrary code that will execute when a rule is selected. However, its primary purpose is to adjust the state of truth in the working memory. One can insert, modify, and retract facts in the working memory. Modifying and retracting facts must be done on fact variable references bound in the LHS. One can explicitly terminate the inference loop with a halt command.

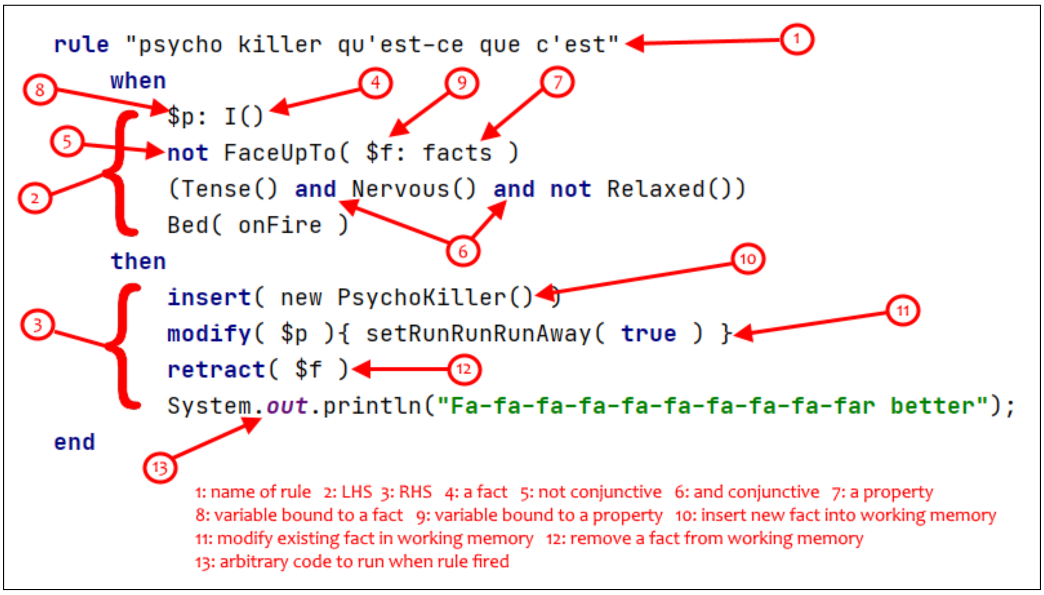


FIGURE 2.3: Drools rule breakdown

An Explanatory Example

Listing 2.1 shows an example of a .drl file taken from the Drools sample code.

```
package org.drools.examples.honestpolitician

import org.drools.examples.honestpolitician.Politician;
import org.drools.examples.honest politician.Hope;

rule ‘‘We have an honest Politician’’
    salience 10
```

```
8           when
9               exists( Politician( honest == true ) )
10          then
11              insertLogical( new Hope() );
12      end
13
14      rule ‘‘Hope Lives’’
15          salience 10
16          when
17              exists( Hope() )
18          then
19              System.out.println(‘‘Hurrah!!! Democracy Lives’’);
20      end
21
22      rule ‘‘Hope is Dead’’
23          when
24              not( Hope() )
25          then
26              System.out.println( ‘‘We are all Doomed!!! Democracy is Dead’’ );
27      end
28
29      rule ‘‘Corrupt the Honest’’
30          when
31              $p : Politician( honest == true )
32              exists( Hope() )
33          then
34              System.out.println( ‘‘I’m an evil corporation and I have corrupted ‘‘ + $p.getName() );
35              modify( $p ) {
36                  setHonest( false )
37              }
38      end
```

LISTING 2.1: Example Drools file.

Listing 2.1 gives the Drools engine instructions on what actions to take when something changes the working memory. This toy example reacts to when the working memory has an honest politician added. It prints a message celebrating the existence of said politician. It then corrupts her and gloats in a message. Finally, it prints a message of despair. The code in Listing 2.1 does the following:

1. On line 1, the package statement identifies the rule file.
2. On lines 3 and 4, the import statements describe which facts are available for use.
3. The “We have an honest Politician” rule on line 6 does the following:
   (a) On line 7, using the salience attribute, the rule is set to be run before other rules with a lower salience.
   (b) On line 10, the rule checks working memory for Politician facts with the honest property equal to true.
   (c) On line 12, if found, then a Hope fact is inserted into the working memory.
4. The “Hope Lives” rule on line 15 does the following:
   (a) Line 18 check if any Hope facts exist.
   (b) On line 20, if found, it prints a message.
5. The “Hope is Dead” rule on line 23 does the following:
   (a) On line 25, it checks that no Hope facts exist.
   (b) On line 27, if it finds no facts, then it prints a message.

6. the "Corrupt the Honest" rule on line 30 does the following:
   (a) Line 32 checks for any Politician facts with the honest property equal to true and sets them to the variable $p.
   (b) Line 33 checks if any Hope facts exist.
   (c) If both Hope and Politician facts are found, on line 35, it prints a message including the $p variables name.
   (d) On lines 36 to 38, it modifies the fact in working memory represented by $p to change its honest property.

2.2 Projectional Editing

2.2.1 What is Projectional Editing?

When talking about projectional editing, we are mostly talking in the domain of Metaprogramming. Usually, when we talk about software development, the programmer or developer creates the program and the user who uses it. In metaprogramming, we are talking about the development of languages. Here the developer could refer to both the creator and the user. We will distinguish these two roles in this paper by referring to the creators of the languages as Language Engineers. Thus when we refer to developers in this paper, we mean users of the language.

Traditionally developers write code with text editors or integrated development environments (IDE), which adjust the concrete syntax and allows a parser to create the abstract syntax tree. A projectional editor, Inverts this relationship, as a developer edits the abstract syntax tree and allows the IDE to project the concrete syntax.

Parser Based Editing

In a traditional parser-based development workflow, a program is defined using text and edited with a text editor. Because it is text-based, the notation of the language is limited to text. A grammar is a definition of a programming language's formal syntactical rules or concrete syntax. One derives the lexer and parser from the grammar. The lexer will turn text passed into a text buffer into tokens. A parser then validates that these tokens, the words of the language, are syntactically correct. The parser then constructs a concrete syntax tree and an abstract syntax tree (AST).

An AST is a tree structure that represents the semantic meaning of the source code, stripped of all the syntactic details. The parser will carry out some of the name resolutions needed to ensure that the tree represents the references expressed within the source code. These references turn the tree into a graph.

Compilers use the AST to do subsequent processing, such as linking, transformation, analysis, and type checking. Modern IDEs, in the background, also parse the code it is displaying to create an AST to offer relevant coding assistance. This assistance is appreciated as without IDE help learning the concrete syntax of non-trivial languages is error-prone. Exploratory programming is laborious if one has to wait until compilation to discover mistakes.

Projectional Definition

In the projectional editing paradigm, a semantic model represents the program. This model requires projectional editing tools to be read and edited.

A projectional editor does not parse any text. In its place, a developer reads and edits a representation of the AST through a projected notation. Her editing gestures immediately and directly manipulate the AST. This editing takes place within predefined and fixed templates called editors.

The principle of projectional editing is familiar to those that use visual programming, like Scratch or Blockly, or graphical modelling tools, such as MetaEdit+. These tools do not parse pixels to generate their AST. Instead, they project the underlying models/programs in a view. They store the model/AST in a custom format rather than its plain text equivalent in a traditional programming language.

Projectional editing is the generalization of this idea, with the ability to render multiple representations of the program with a wide range of notation styles.

The projection may sometimes seem like a text editor. However, this is just acrobatics by the language engineer designing an editor to help developers from traditional text-based languages feel comfortable. The text is just another type of projection of the AST. It also may be any other notation that can represent the semantic meaning of the code, such as formulas, graphs, or images. Projections are not just the notation but also how the user interacts with the projection. In this sense, the definition of the projections and the IDE/UI overlap.

2.2.2 What is it Not?

Projectional editing does not have clearly defined boundaries. In this paper, we exclude the following types of tools that sometimes get associated with projectional editing.

A Venn diagram of Model-Based Software Engineering (MBSE) and projectional editing would have a significant overlap. Here we will not be looking at tools that build code from UML or other MBSE or Model-Driven Engineering (MDE) tools.

Another area mistakenly grouped with projectional editing is Low-code software development environments. These, however, are only tangentially related.

Most confusing is projectional editing when used to refer to a methodology of product line differentiation in code bases. In addition to having the same name, one of the top products for this product line technique is called PEoPL. PEoPL uses MPS for development. Thus, this product is a projectional editor (the paradigm) for product line projectional editing (the methodology).

How Projectional Editing Works

As shown in figure 2.4, a projectional editor has a model or an AST. It renders a presentation of the model as a projection. The developer performs actions on the projection. Every user editing action maps directly to a change in the AST.

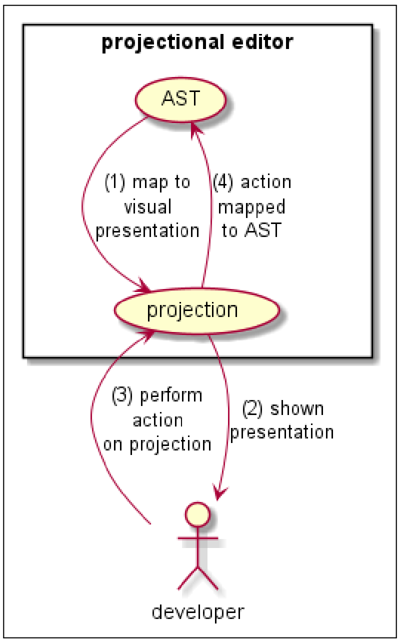


FIGURE 2.4: Projectional editing loop. TODO: comparison with Parsing

To perform the above two things have to be defined by language engineers: The Meta-Model and the editor.

The meta-model, analogous to the abstract syntax, describes the node concepts and connections used to build the hierarchical structure that is the AST. This hierarchy can have references to nodes in other branches, so it is a graph although named a tree. The AST is stored independently of the concrete syntax, often using a database, XML or a proprietary file format. Rules of the meta-model, such as type systems or scoping rules, must also be described.

Projectional editors avoid the grammars and parsers that define the concrete and abstract syntax in a traditional text-based language. In text-based languages, the concrete is transformed to the abstract with parsing. In projectional editing, the abstract is transformed to concrete using a projection engine that uses projection rules.

Editors combine the projection rules and the gestures or actions that will create a change request to the AST. They are analogous to a concrete syntax.

One of the actions can be typing text. However, every string is recognized when entered. Thus, there is no tokenizing. Text enters into the templates defined by the editor and a newly derived projection displays the adjusted underlying AST to the developer.

The projection uses graphical elements to represent the representation. Although often appearing textual, each of the text elements are references to nodes in the AST.

Developers can only interact with the editor via the rigidly controlled code completion menus or gestures and actions. She builds the AST directly from each interaction she has with the editor.

Nodes are instances of the concepts defined in the meta-model. Each node has a unique id and points to its defining concept. It is unambiguous. References are first-class and defined by the id rather than resolved by name, as in parser-based languages.

Disambiguation happens at the time of input, as the developer chooses from limited legal inputs.

The separation of the abstract and concrete allows the language engineer to implement multiple projections of the same model, using different notations, each node of the AST taking having the design she envisions. The pattern used for projection is similar to MVC, so multiple views of the program can be visible and updateable simultaneously.

Graphical modelling tools, for example for UML modelling, could be seen as specialized implementations of projectional editing. These modelling tools do not store pictures of the UML diagrams and then parse them to create an AST. Instead, they store the model, often with extra information about the visual layout, and the image of the UML is projected to the modeller to edit. Projectional editing generalizes this approach to projecting any notation defined by the language engineer.

2.2.3 History of Projectional Editing

Here follows an incomplete and inconsistent history of projectional languages.

In the '70s and early '80s, researchers created several applications for research into the realm of structured editors. Some examples were: MENTOR[13], Incremental Programming Environment[14], GANDALF[15], Cornell Program Synthesizer[16], and Synthesizer Generator[17]. These language-based program editors could force syntactically correct programs through the knowledge of the language. These were the precursors to the modern projectional editors. They worked by providing templates for each abstract computational unit of the language. First, one would choose the concept and then fill out the placeholders.

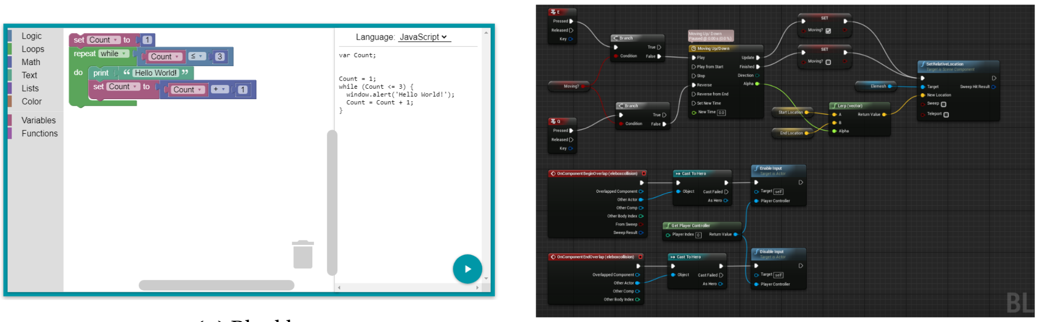
These tools were not good at editing textual notations, which led to a poor user experience. When they attempted to fix this, for example, in the Synthesizer Generator, they reintroduced parsing to parts, which took away many of the advantages of direct editing of the AST.

In the late '90s and early '00s, the first forays into commercializing a more generalized version of structured editors, the projectional editors, began. First, Intentional Domain Workbench, inspired by Charles Simonyi's 1995 essay "The Death of Computer Languages, The Birth of Intentional Programming"[18] the IDW was the product of the company Simonyi founded in 2002, Intentional Software. The Intentional Programming paradigm spotlighted the projectional editing domain, taking it out of the universities and into practice. However, as it was a closed sourced and expensive product, thus not many papers were written about it. In 2017, as part of an "acquihire", Microsoft bought Intentional Software for its employees and let the product die.

Inspired by a call to action for Language orientated programming[19] JetBrains embarked in 2004 on a mission to build a product to fulfil that ideal. Meta Programming System (MPS) was the outcome of that journey. Language engineers created the languages mbeddr, PEoPL, and Realaxy using the MPS platform. It currently has an active community of developers and projects both in academia and in the commercial world. We chose this tool to be the basis of our projectional editing experiments. We will talk about it at greater length in section 2.3.

The last decade has produced a few smaller projectional workbenches. There are a few open-source, small team projectional projects. In 2013 several projectional language workbenches joined MPS in the Language Workbench challenge[20]. These included Más, a web-based projectional editor, which is no longer with us[21]. Whole Platform[22] is a projectional language workbench plugin for Eclipse. Cedalion[23] provides another projectional IDE, specializing in internal DSLs.

More recently there have been some interesting products that intersect the projectional domain. Deuce[24] and Gentleman[25] are two recent projection editors that have recently emerged from academia. The final two mentions in our incomplete history are a little out of left-field. Google's Blockly[26] is a tool for making structural editor languages, but only in a block format. This can create languages similar in style to the scratch language. Blueprint visual scripting, a part of the Unreal Engine, is a visual programming languages for building concepts such as levels or game assets. Examples of Blockly and Blueprint can be seen in figures 2.5a and 2.5b respectively.



(A) Blockly

(B) Blueprint

FIGURE 2.5: Leftfield projectional editors

What Advantages Does Projectional Editing Bring?

Projectional editing gives advantages both to the language engineer and the program developers. There is a lot of crossover and repetition between papers written on projectional editing as to the advantages it brings. To that end, what follows is a synthesis of several papers as to the advantages that projectional editing claims. Rather than attributing each advantage to each paper, we have made a reference table of papers proclaiming said advantage. This is table 2.2.

Exploratory programming As with their progenitors, syntax-directed editors, modern projectional editors help guide a developer unfamiliar with a language. With their rigid syntax and predefined layout, the editors only allow editing within specific cells of the editor. This template style means that the developer does not have to worry about the significance of spacing or indentation. Minutiae of syntactic adornments, such as statement ending semi-colons or enclosing matched brackets, are also not interfering with her exploration of the language space.

When creating code, the editor only presents the developer with legal options within the current context. As the projection is context-aware, with relevant actions and options suggested and irrelevant ones removed. Thus, it is easier for the developer to explore what

Advantage	#	Paper(s)
Exploratory programming	5	[27–31]
Correctness-by-construction	7	[27, 32–36]
Rich notation	22	[18, 27, 29, 30, 35–52]
Mixed notation	8	[28–30, 38–42]
Multiple views	9	[27, 29, 36, 38, 39, 41, 43, 47]
Language composition	23	[18, 29, 30, 32, 33, 35, 36, 38–42, 47, 48, 51–59]
IDE functionality	4	[27, 39, 41]
Language evolution	1	[60]
Ancillary data	5	[29, 39, 52, 58, 59]

TABLE 2.2: Papers describing advantages

the language allows her to choose. Intelligent code completion does not have to be limited to single nodes. Inserting whole subtrees allows the developer to explore the larger structures of the language.

Correctness-by-construction A projectional editor prevents her from writing syntactically incorrect code by controlling the interaction between the developer and the AST. The whole class of syntactical errors is made impossible, with the developer relieved to think about special characters and layout. Typing and scoping errors are removed by only allowing validly typed and scoped options for the developer.

The developer can only select statements that are legal in the context of the location within the AST. Code does not have to be disambiguated, as this happens at the time of entry by the developer. If multiple items share the same notation in the editor, the developer chooses the relevant item, thus resolving the ambiguity to what she means rather than what the parser thinks she means.

Rich notation Constraints associated with textual parsing do not affect the choice of created projections. This freedom opens up diverse otherwise tricky or impossible to parse notations. Examples include tabular, mathematical expressions and symbols, diagrams, trees, images, forms, prose, sub- and superscript. Any visual form or shape that can map onto the AST can represent the program in an editor.

With these notations, one can better reflect the semantics of the program domain, which should aid comprehension. Mathematics has a rich history of use of notation. When writing a DSL for the Mathematics domain, the domain experts can interact with it in the centuries-old language of their domain.

Of course, the projections can also be projections of text. Textual projections are often the appropriate projection type if the developer interacting with the language's domain expertise is parser-based languages.

Mixed notation As no parsing is required and ambiguity is not an issue for the underlying AST, it is straightforward to combine different forms of rich notation. With all notations working on the same editor infrastructure, embedding mathematic symbols within textual projections, within tables within graphical representations is a simple coding pattern.

Multiple views With the AST being the stored artefact rather than the notation, projectional editing allows the language engineer to define multiple views on the same model optimised for different tasks. In Software architecture, one presents different views to different stakeholders based on their interests. Similarly, projectional editing can present experts with various domain expertise views on the model that reflect their needs. A developer can switch between different node projections within a larger projection to find the one that best suits their current task.

Because the architecture of a projectional editor follows the principles of model-view-controller, it is possible to have multiple simultaneous views of the model. These multiple views allow the developer to update a projection optimised for writing and immediately see its effect in a projection optimised for understanding.

Language composition Parser-based languages can support some modularization and composition, but a projectional editor allows easy and extensive modular language extension and composition. This ability results from the disambiguation of the nodes of an AST at the time of entry. If two items with the same syntax are available at the same place, then the user will choose the one they require, and therefore the node has an explicitly chosen meaning.

The composition of independently developed languages does not suffer from the syntactic or keyword clashes they would in two grammar defined languages. Because of the lack of ambiguity, every node referencing the concept that defines it, these languages, when put together, will not have structural or syntactic issues.

Language composition can involve extending an existing language or embedding other languages in a host language without modifying its definition. The ease of composition and extensions allows building more significant languages out of smaller modules.

IDE functionality Developers in mature languages are used to the functionality of mature IDEs. These functionalities include syntax highlighting, intelligent code completion or suggestion, and static analysis for errors and validation. As projectional languages store the AST rather than the concrete syntax, they require an IDE to edit. Because of this, when a language engineer designs the language, she also has to design the IDE.

A projection always knows its context because it comes from the AST. When the editor already knows the meaning of the node it represents, syntax highlighting is simple. Knowing its context makes it much simpler to suggest intelligent code completions.

Always having a complete AST makes it much easier to validate scope, typing and other hard to implement code validators.

Language evolution Parsing complicates the evolution of languages. For example, adding a new reserved word is difficult without breaking existing code. Extending a language with new capabilities and syntax in projectional editing is simple. If the change is syntactic, then the language engineer has to update an editor. If there is a semantic change, then the language engineer can write a migration in the language to transform a node of one concept to a different type, and the developer would have to run that migration on their code.

Ancillary data Data added to nodes can augment the AST. This data is useful for tasks such as documentation, requirements traceability and product line feature dependencies.

What are the Disadvantages of Projectional Editing?

Whilst fewer papers proclaim the disadvantages of projectional editing, we repeated the approach of the previous section. Thus, we have synthesised the disadvantages from papers in the following sections and listed citations for these ideas in table 2.3.

We do not consider that the dearth of disadvantages discussed as evidence of projectional editing's superiority. Our best guess is that those who do not find projectional editing useful do not write papers about it.

Disadvantage	#	Paper(s)
Low adoption	4	[36, 40, 49, 51]
Unnatural user experience	11	[30, 35, 36, 38, 39, 41, 47, 49, 51, 57, 60]
Ambiguous syntax	1	[42]
Inflexibility	2	[30, 38]
lack of integration with text ecosystem	5	[30, 38, 49, 57]
Learning curve	5	[30, 39, 40, 48, 57, 61]
Vendor lockin	2	[39, 41, 62]

TABLE 2.3: Papers describing projectional editing disadvantages

Lack of adoption The ideas that proceeded projectional editing - the structured or syntax-directed editor - have been around since the early 1970s yet have failed to be adopted widely. This argument is a bit of a tautological one, as the low adoption is perhaps an outcome of the other disadvantages of projectional editing. However, low adoption can lead to a self-reinforcing process, where lack of adoption prevents further adoption.

Inconvenient or unnatural editing Early attempts at projectional editing presented an inconvenient and unnatural user experience when coding. These usability challenges, exemplified by the tedious manner of entering code as per the tree's order, compare poorly to parser-based languages.

This lousy reputation continues, despite massive improvements in projectional editors. Whilst there is no debate that projectional editing feels different, some question whether this inconvenience is an intrinsic property or a result of developers, through years of experience, being used to text-based programming.

Modern projectional editors, when using a textual projection, face an "uncanny valley" issue. Whilst trying to simulate a text editor, the developers start to expect all of the functionality of the text-based IDEs. This expectation is a particularly weak spot, especially regarding granularity and restrictions of cursor movement, insertion, deletion, selection, copy and paste, and other interactions with the text.

Ambiguous syntax One of the selling points of projectional editing, especially concerning language composition, is that there can be no ambiguous syntax. While this may be true for the AST, it is not so for the developer who reads this code on the screen. If one combined Drools and Basic rather than Java, the developer might become confused about which language the “Then” keyword refers to when she read it. Thus writing ambiguity is replaced by reading ambiguity.

Inflexibility A developer using a projectional editor has no flexibility in code layout. They may feel they require this for enhanced readability. The flexibility of the layout is entirely in the hands of the language engineer when she determines the projection rules.

Integration with the text based world Projectional editors do not store the definition of the program in the form of a plain-text implementation in the concrete syntax. Instead, the AST is stored and serialized in a format that is not human readable, such as XML.

This different format of program storage leads to an issue with integration with the text-based ecosystem. This ecosystem is extensive, as text-based coding has been popular since the 60s. Two notable examples are text diffing, especially where branch merging is concerned and code sharing. The diffing issue within projectional editing tools is solved. However, as code-bases often span multiple programming languages and tools, the difficulty of integrating projectional diffing into the software development workflows is still a real problem.

Textual source code can be shared simply by email or on websites. This sharing, however, is not easy with projectional code.

Learning curve For the language engineer, the necessity to develop an editor with a good user experience is much harder work than defining a grammar for a parsed language. The learning curve for the language engineer is significant, as, by default, she has to think also of the IDE development.

For the developer, especially one with an extensive text-based experience, the different editing style takes some getting used to.

Vendor lock-in The nature of projectional editing is that what one edits is a projection of the AST, and therefore an IDE is needed to do the projecting and language definition. Organisations thinking of going the projectional route for their DSLs may fear being locked into a specific concept implementation. To be able to use previously developed languages would require using the same toolset. Changing to a different toolset for language design would require a significant re-skilling effort.

2.3 What is MPS?

Language workbenches (LWB) are a tool to help language engineers create languages, particularly domain-specific languages (DSL). Fowler[63] popularised the term LWB in a 2005 article.

Meta Programming System (MPS) is an open-source LWB that assists in the creation of Projectional languages. It started in 2003 by JetBrains and was introduced to the world in Sergey Dmitriev's 2004 Paper "Language Orientated Programming: the Next Programming Paradigm"[19].

As discussed in section 2.2.1, When creating a projectional language, one has to define the language and how one interacts with it. In MPS, the language engineer defines languages, including their interactions. Developers create programs using these languages. The language engineer can extend languages. The developer can mix the languages she uses.

The following is an overview of how MPS implements the ideal of a projectional language. It is also the structure of this section:

- Abstract syntax
  - Structure
  - Behaviors
  - Constraints
  - Type system
- Concrete syntax
  - Editors
  - Intentions
- Generators
  - Model-to-Model
  - Model-to-Text

MPS defines the different aspects of the language definitions with small, declarative DSLs. These are bundled together into what they term Aspects.

2.3.1 Abstract Syntax: Structure

Structure is what determines the abstract syntax of a language. The most important item available in a Structure Aspect is the Concept. Instances of concepts are called nodes. With these nodes, the developers construct their programs. When referring to a program in MPS, we are talking about its stored abstract syntax tree (AST).

In principle, a concept contains three types of things:

1. Properties: these primitives are integer, boolean, string, or enum items and are similar to leaf values.

2. Children: these are other concepts, or collections of them, similar to subtrees.

3. References: these are relationships with other nodes in the AST. These turn the tree into a graph.

Concepts follow some object orientated (OO) traits, such as subtype, being abstract, and implementing interfaces.

One of these Concepts must be a root node. Otherwise, there is nowhere for a program to start.

Other items available in the Structure Aspect are the Interface Concept, the Enumeration, the Constrained Data Type, and the Primitive Datatype. [1]

Thus, the structure aspect defines how the AST can be structured.

Figure 2.6 shows a concept with three children that implements two interfaces.

```
concept RuleStatement extends    BaseConcept
                      implements INamedConcept
                                 IFileLevelStatement

  instance can be root: false
  alias: rule
  short description: rule

  properties:
  << ... >>

  children:
  attributes  : RuleAttributes[1]
  conditions  : AbstractCondition[0..n]
  outcomes    : StatementList[1]

  references:
  << ... >>
```

FIGURE 2.6: Concept example

2.3.2 Abstract Syntax: Behaviors

[2] OO design usually bundles together data and methods that can act on that data. Concepts are analogous to the data part of this equation. Behavior fills the role of the methods in the OO analogy, defining the functionality called from instantiated nodes and static

[1] At the time of writing, we are unaware as to whether Data Type and Datatype are semantically different or if it is a style choice.

[2] When referring to its use in the MPS workbench, consistent with their use, we use the American spelling - Behavior.

methods called from the Concept. The Constructor is a specialised method in a Behavior, filling the same role as a constructor in OO.

The methods have public, private, or protected visibility. If the Concept to which the behavior refers is abstract, the behavior itself can contain abstract methods. Abstraction, variable visibility and inheritance allow a sort of polymorphism. If a virtual method is declared, then it can be called polymorphically.

Figure 2.7 shows a constructor added to a Concept to initialize its children. It has a method to allow other nodes to interrogate the condition of it having attributes.

```
concept behavior RuleAttributes {

  constructor {
    this.salience = new node<SalienceAttribute>();
    this.salience.salience = new node<IntegerConstant>();
    this.salience.salience:IntegerConstant.value = 0;
    this.noloop = new node<NoLoopAttribute>();
  }

  public boolean hasAttibutes() {
    return this.salience.visible || this.noloop.visible;
  }

}
```

FIGURE 2.7: Behavior example

2.3.3 Abstract Syntax: Constraints

The constraints aspect adds further structural constraints to concepts. These constraints are especially used to define scope. Other constraints including allowing if a node can be a child, a parent, or an ancestor of this node. the constraints aspect also allows preventing badly formed properties, children or references.

Figure 2.8 shows an example of a scope restraint that only allows local variables that are declared within the same rule or global variables declared in the same file.

2.3.4 Abstract Syntax: Type System

The Typesystem aspect and the constraints aspect together represent the static semantics of the language. This aspect is for the computation and evaluation of types of variables, expressions and statements.

Rules that are available to calculate and enforce the type system include inference, subtyping, comparison and substitute type rules.

Figure 2.9 shows an inference rule that ensures that the calculated type of the import statement matched that of its child called type.

```
concepts constraints RuleVariableRef {
  can be child <none>

  can be parent <none>

  can be ancestor <none>

  instance icon <none>

  <<property constraints>>

  link {target}
    referent set handler <none>
    scope (referenceNode, contextNode, containmentLink, position, linkTarget)->Scope {
      node<RuleStatement> rule = contextNode.ancestor<concept = RuleStatement, +>;

      nlist<RuleVariable> localVars = rule.descendants<concept = RuleVariable>;

      sequence<node<RuleVariable>> globalVars = rule.containingRoot.descendants<concept = GlobalStatement, +>;

      return ListScope.forNamedElements(localVars.concat(globalVars));
    }
    <no presentation (deprecated)>


  default scope <no default scope>
}
```

FIGURE 2.8: Constraint example

```
inference rule typeof_ImportStatement {
  applicable for concept = ImportStatement as importStatement
  applicable always
  overrides false

  do {
    typeof(importStatement) :==: typeof(importStatement.type);
  }
}
```

FIGURE 2.9: Typesystem example

2.3.5 Concrete Syntax: Editors

Editors define the notation of the nodes. In effect, it is the user interface of the language, how the AST is projected to the developer. An editor is a swing panel that renders a tree of editor cells. A concept can have multiple editors, thus offering multiple views on it.

The definition of the options available to the developers through menus also happens within the Editor Aspect. The choice the developer makes transforms the existing AST.

Additionally, the behaviour of interactions can be defined, such as what will happen to the AST when a particular keypress or editor action occurs at a particular location.

Figure 2.10 shows a component with a projection for the Concept shown in figure 2.6.

2.3.6 Concrete Syntax: Intentions

In projectional editing, the IDE is a part of the concrete syntax. Intentions make context-aware suggestions for automatic changes to the program to the developer. Figure 2.11

```
editor component rulestatement
  overrides:

  applicable concept:
    RuleStatement

  component cell layout:
    [/
       [> rule " { name } " <]
       ?[> ---> % attributes % <]
       [> ---> when <]
       [> ---> ---> (/ % conditions %        /) <]
                      /empty cell:
       [> ---> then <]
       [> ---> ---> % outcomes % <]
       end

    /]
```

FIGURE 2.10: Editor example

shows an intention that allows the developer to add, remove or edit a property based on its current value.

```
intention addSalienceRule for concept RuleStatement {
  error intention : false
  available in child nodes : true
  child filter :

  description(node, editorContext)->string {
    node.attributes.salience.visible ? "Remove Salience Attribute" : "Add Salience Attribute";
  }



  execute(node, editorContext)->void {
    if (node.attributes.salience.visible) {
      node.attributes.salience.visible = false;
    } else {
      node.attributes.salience.visible = true;
      node.attributes.salience.select[in: editorContext, cell: FIRST_EDITABLE];
    }
  }
}
```

FIGURE 2.11: Intention example

2.3.7 Generators: Model-to-Text

Whilst designing a language is nice, it has to be able to do something. Without doing so, it has no semantic meaning. It is possible to create interpreters that can use the AST generated by MPS. However, the most common modality for MPS is to generate an output that gets compiled and run by commonly known environments. The output stage of generation is called TextGen. It defines how a node becomes runnable code in plain text.

2.3.8 Generators: Model-to-Model

Base level languages will have text generation. Most DSLs will perform model-to-model conversions, eventually converting to a base language. These intermediate stages are known as Generator Aspects. They transform code written in one language to another.

In MPS, a Concept can have multiple generators aimed at different base level languages, such as Java (using MPS BaseLanguage), C (using mbeddr) or XML.

Chapter 3

Methods

To answer the research questions from section 1.2, we formed three approaches. The question

3.1 Method: Systematic Review

To answer the first Research Question, "what is the current state of Projectional Editing?", we conducted a systematic literature review. Hereafter, we describe the method we undertook.

To carry out this review we followed Kitchenham's[64] advice on systematic review protocol validation, (see appendix A for the exact checklist we used).

3.1.1 Motivation

The motivation that preceded this research was a requirement to understand if projectional editing was an idea that was worth investigating. Through our background research we saw an interest in the precursors to projectional editing in the late 70's through to the mid 80's. This seemed to be abandoned until the mid 90's following Charles Simonyi's treaties on Intentional programming. This did not lead to a swell in academic research as his companies product, Intentional Domain Workbench was a closed commercial product. There seemed to be a burst of academic interest after the release of JetBrain's Open-Source Meta Programming System (MPS) in the late 2000's.

Is there a need for a study of this topic? We believe, at least in the microcosm of this master's project it is helpful to know whether we are researching in an area that is dying of vibrant.

For the wider community, there does not seem to be any systematic reviews specifically about projectional editing, let alone recently. This study is not extending any previous Systematic Review as, although there were literature surveys and mapping studies covering some adjacent fields, no SLRs were found. Thus we believe it may be useful for those in the language engineering research community to bring together all current research in the area of projectional editing in one place.

3.1.2 Research Question

This paper we only have one research question to synthesize the findings of scientific papers towards. This is “What is the current state of Projectional Editing?”

This question for us can be broke down into:

- **Sub Question 1** “Is there current research in the area of projectional editing?”
- **Sub Question 2** “What tools are currently being used for research?”
- **Sub Question 3** “What is the sentiment in papers currently discussing projectional editing?”

3.1.3 Search Strategy

The search process is automated as SLRs require a high level of completeness, which cannot be effectively achieved manually. Our first major decision was whether to engage in creating a quasi-gold standard as advised by Zhang[65]. Zhang noted that the ad-hoc nature of search strategies in SLRs has limitations. We executed a preliminary ad-hoc search to try and ascertain the extent of the research space. Upon satisfying ourselves that it was small enough, we rejected the Quasi-Gold standard as overkill for our requirements.

The search terms we landed on were as follows:

```
“PROJECTIONAL EDITING”

OR

“PROJECTIONAL EDITOR”
```

This is to be adjusted to fit the query syntax of the various search engines.

As most Research Search engines offer the option of date ranges, and to save the effort of excluding later we also used the date range to eliminate unnecessary paper at the automated search stage. Our research question we are specifically looking at the current state of projectional editing. A design decision of many research search engines is that date ranges can often only be defined in whole years. When designing our search strategy, it was near the beginning of 2021, and thus we feared that this would be too small a search space, thus we set our date range to be from the beginning of 2020 to present. For the sake of reproducibility, it is advised to remove any papers after 31st July 2021.

The Search Engines used are shown in table 3.1.

ACM digital library	Google Scholar
BASE	CORE
IEEE Xplore	ISI Web of Science
Microsoft Academic	Science.gov
Wiley InterScience	SCOPUS
Semantic Scholar	SpringerLink

TABLE 3.1: Search engines used

Once we have filtered the automated search through the criteria of the selection stage, we will use that as our starting set for snowballing. Our filtering will be done before the quality of the papers has been assessed, as we feel that excluding papers on quality of primary study issues may artificially limit the network of potential papers. Our snowballing procedure shall follow the advice of Wohin[66]. This is the idea of using the reference lists from our start set and applying the same selection criteria to these.

Where possible we will get the forward snowballing papers from the "cited by" functionality of Google Scholar. Because of the range of the search being to present, all papers that cite the target paper will fall within our criteria. For backward snowballing we will manually filter the bibliography section of the selected papers, selecting any paper published in 2020 or 2021

After gathering all the papers from the forward and backward snowballing we will again apply the selection criteria. This process will iterate recursively until no new papers are found. All of the papers not excluded in each iteration will be the basis for the quality of primary studies filtering stage.

After the final iteration, as a final step the selected papers will have a deeper scan. This is to verify our initial scan that the papers met our inclusion criteria, before moving on to the quality assessment.

3.1.4 Study Selection

The inclusion criteria are:

- Studies are about or mention projectional editing or one of it's synonyms
- It is published in during the period 20202021

The exclusion criteria are:

- Books and grey literature
- not English
- no full text available
- papers with serious issues with grammar or vocabulary
- not a previously selected paper

- not a paper about a previously reported on study

If multiple papers look at the same study with different approaches, then the data will be aggregated in the synthesis stage.

As a lone researcher, we must be aware of bias in positively including relevant papers and excluding irrelevant papers. We will follow Kitchenham's suggestions to overcome such bias:

- Test-retest
  - We will assess the papers once (on title abstract and keywords) against the inclusion and exclusion criteria.
  - Save all the suggested results
  - Assess the papers again three days later in a different order to the first
- If there are disagreements, we will use Cohen's Kappa agreement statistic[67] to see if the process needs to be refined.

If our searches appear to be too large for a lone researcher, we will turn to text mining. We will be cautious to use this. O'Mara-Eves et al.'s systematic review of text mining in systematic reviews[68], recommends that this can be used for prioritization, but finds that for exclusion screening, although promising, it is not yet proven.

An SLR is interested in studies rather than papers. There is a many-to-many relation between papers and studies. We will review the selected papers to note when this has happened in our results to make sure studies do not get over or undercounted.

3.1.5 Quality of Primary Studies

To discover explanatory reasons for why there may be differences in study results, and to weigh how valuable specific studies are, we will assess the quality of the selected studies.

To try and avoid a Results Section bias we will be operating a results-blind quality assessment. Study quality will be based on the methods section of the papers only. This bias is threatened because results are summarized in the abstract. The study quality will not be measured until after the selection process is complete, though it will, in part be carried out before the selection re-test. list of papers will be randomly sorted before assessing for quality.

For EBM studies there are some well-known hierarchies of evidence for study quality thresholds such as the CRD Hierarchy of Evidence[69]. Kitchenham in, Procedures for Performing Systematic Reviews[70], suggests the hierarchy shown in table 3.2

Rank	Description
1	Evidence obtained from at least one properly designed randomized controlled trial
2	Evidence obtained from well-designed pseudo-randomized controlled trials (i.e. non-random allocation to treatment)
3-1	Evidence obtained from comparative studies with concurrent controls and allocation not randomized, cohort studies, case-control studies or interrupted time series with a control group
3-2	Evidence obtained from comparative studies with historical control, two or more single-arm studies, or interrupted time series without a parallel control group
4-1	Evidence obtained from a randomized experiment performed in an artificial setting
4-2	Evidence obtained from case series, either post-test or pre-test/post-test
4-3	Evidence obtained from a quasi-random experiment performed in an artificial setting
5	Evidence obtained from expert opinion based on theory or consensus

TABLE 3.2: Study design hierarchy for software engineering

These different types of study have different quality assessment criteria. As the nature of our research questions is not likely to attract randomized or pseudo-randomized controlled trials or experiments, our quality assessment checklists are created with comparative studies and case series in mind. To assess the strength of each primary study we used the checklists shown in appendix C. The checklists were based on a subset of the questions suggested in Keele's guidelines on SLRs[71], which in turn extracted questions from previous mostly medical systematic reviews. Where necessary the selected questions were modified for software engineering.

These checklists are addressed toward general research. In Software Engineering many studies that fall under what Gregor[72], in "A Taxonomy of Theory Types in Information Systems Research" calls "Type V: Theory for Design and Action". The checklists do not address this type of research well. On investigation into how others SLRs conduct quality assessment we did not find a solution to this issue. Therefore we will continue with the checklists as in the appendix, using the checklists for Case Study for Type V research papers. We will take this into account before dismissing results of this type on basis of their quality score.

As this study will be carried out by a lone researcher, there is no need to have a process for disagreements. To check on the bias, several papers will be randomly selected and assessed using the checklist by the academic and the daily supervisors. Should there be a high disagreement the design of the checklist will be revisited.

The quality checklist is trying to weed out the biases of selection, performance, detection, and exclusion, as well as other threats to the validity of the studies under test. Validity issues can occur during the design, operation, analysis, or conclusion of an empirical study.

3.1.6 Data Extraction

No data extraction will be necessary for the first sub-question, "Is there current research in the area of projectional editing?". The fact of the existence of papers that have been verified to be primary studies either into projectional editing theory or practical use of it will be enough to answer the question.

For the question of "What tools are currently being used for research?", we shall note each tool discussed specifically with regards to the study being carried out.

Finally, for the sentiment we shall pass each paragraph of the introduction, the discussion and the conclusion through a sentiment analyser and, if the paragraph is pertinent to projectional editing, we will not it's sentiment score. The sentiment analysis tool we shall use is Microsoft Azure Cognitive Services Text Analytics. The code to carry out this task is shown in listing 3.1

```
from azure.core.credentials import AzureKeyCredential
from azure.ai.textanalytics import TextAnalyticsClient

endpoint = "REPLACE_WITH_CORRECT_ENDPOINT"
key = "REPLACE_WITH_CORRECT_KEY"

text_analytics_client = TextAnalyticsClient(endpoint=endpoint, credential=
    AzureKeyCredential(key))

inputfiles = [[ARRAY_OF_FILES_TO_BE_ANALYSED]]

with open('/content/sample_data/sentiment/output_all.txt','a') as outf:
  for sections in inputfiles:
    for section in sections:
      print("Section: {}".format(section),file=outf)
      f = open('/content/sample_data/sentiment/'+section)
      content = f.readlines()
      # for brevity an optimization to deal with 10 document limit is removed
      if len(content) != 0:
        result = text_analytics_client.analyze_sentiment(content,
            show_opinion_mining=True)
        docs = [doc for doc in result if not doc.is_error]
        for idx, doc in enumerate(docs):
          print("sentiment: {}".format(doc.sentiment),file=outf)
          print("Document text: {}".format(content[idx]),file=outf)
```

LISTING 3.1: Text Analytics code.

#	Data Type	Description	RQ
1	Study ID	Unique identifier for the study	
2	Title of Study	The paper name	
3	Year of Publication	Will be either 2020 or 2021	
4	Author(s) Names	Including affiliation	
5	Source of Study	Name Of Online Database/ Digital Library	
6	Type of Study	Publication/Conference/Workshop/Symposium	
7	Name of Venue	Journal/Conference in which study has been published	
8	Tools in Study	A list of the tools used	RQ 1.2
9	Sentiment	The sentiment scores from appropriate paragraphs	RQ 1.3

TABLE 3.3: Data extraction form

3.1.7 Data Aggregation and Synthesis

As explained by Kitchenham[73], in software engineering, primary studies will tend to be too heterogeneous for any statistical analysis. Synthesizing outcomes from multiple methods will be difficult. Thus our synthesis will take a narrative approach.

Narrative synthesis is telling a story of the who, how, and why of the success or otherwise of the research. For ADR research, focus will be on what will help or hinder the adoption of the implementations. It will also examine how reliable the results are and relationships between the studies.

3.2 Method: Action Research - Drools in MPS

Even though Drools is a relatively small DSL, we did not need to implement all the functionality to answer our questions.

3.2.1 Really Simple Rules Language

As we were new to DSL design and MPS, we first would create a simple approximation of the Drools language to create our first projections. We called this language “Really Simple Rules” (RSR).

File RSR, Like Drools itself, has a File as its root node. The File only contains Facts and Rules.

Fact and FactProperty In Drools, a fact represents a Java Bean with its child properties, which can also have their child properties, ad infinitum. In RSR, we limited properties to allow only boolean values. We decided this because fact selection is a predicate and thus can only return a boolean. By only allowing booleans, we also simplify the operations allowed on the property.

Rule We only simulated the Left Hand Side, or the “When” conditions, of a Drools Rule for the Rules Concept. We believed this would provide us with compelling options for projections and did not want to overcomplicate this first approach.

An RSR Rule consists of a collection of conditions. Should all those conditions return true, then the rule is selected.

Condition A condition operates on one or more FactSelectors. There are four condition types ExistsCondition, NotCondition, AndCondition, and OrCondition. ExistsCondition and NotCondition are unary conditions and evaluate one FactSelector. AndCondition and OrCondition conditions evaluate two FactSelectors.

FactSelector A FactSelector consists of a reference to a Fact and a collection of Predicates. If the Fact exists and all the predicates evaluate to true, then the FactSelector evaluates to true.

Predicate The predicate is an operation on a FactProperty, to which the concept has a reference. As the FactProperty represents a boolean value, the only predicate operations are “Is” and “Not”.

Figure 3.1 shows the Concept hierarch for this straightforward implementation.

We realised this design in MPS. As the aim is to attempt different projections, we did not initially optimise for editing. The structure is as shown in figure 3.2 , and the editors including those shown in figure 3.3.

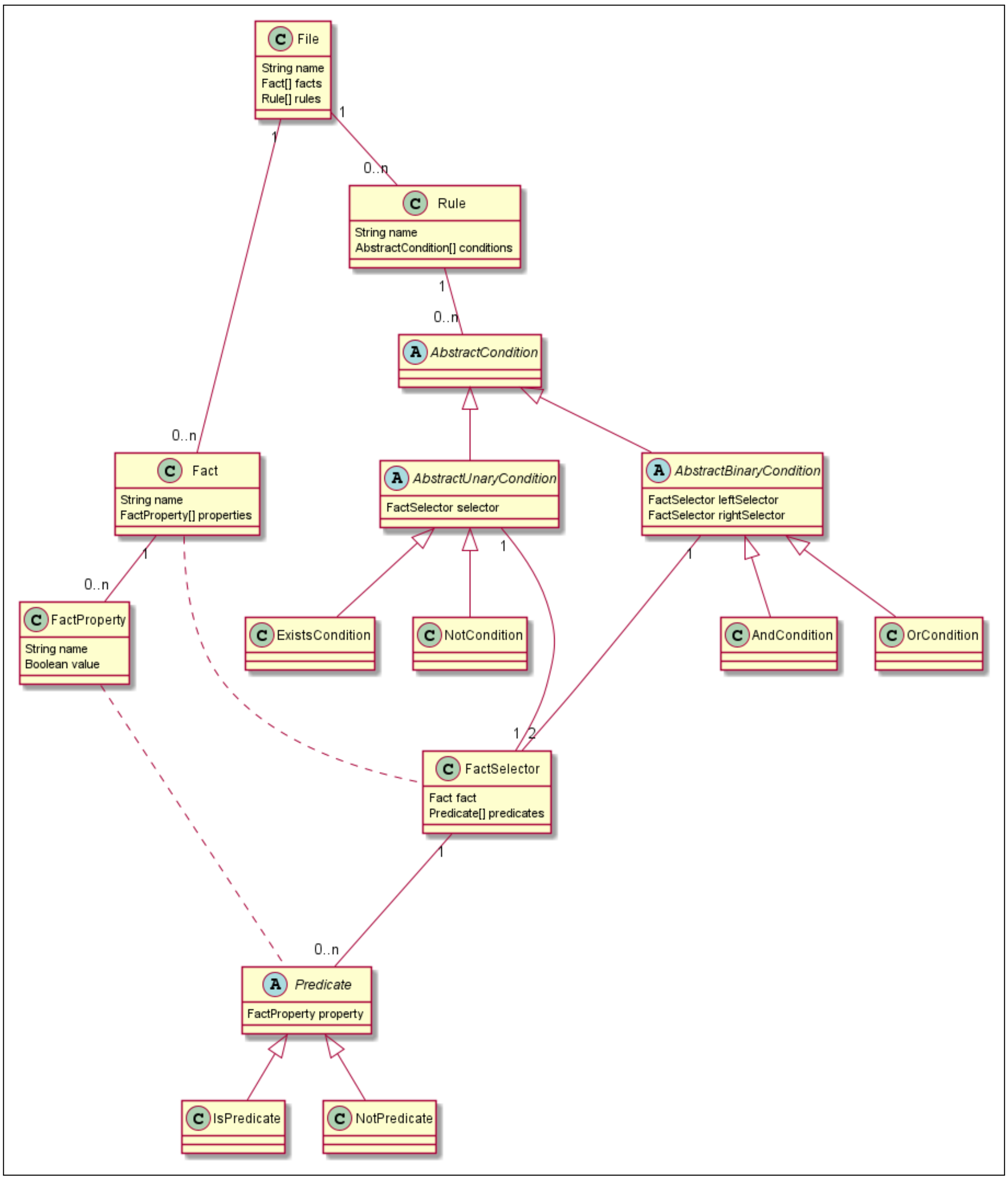


FIGURE 3.1: RSR concept hierarchy

Part of the research question is using projections for reasoning about large files. In order to answer this, we needed to simulate a large file. To do this, we had to enter a large number of rules. As this becomes tedious, we added some editing aids, including substitute menus, to speed up the entry of conditions, as shown in figure 3.4.

This image shows that we originally had to select an ExistsCondition Concept and select the Fact for the condition. After adding the substitute menu, We could immediately select the Fact we wanted, and the ExistsCondition would automatically wrap it with an ExistsCondition node.

We also added some intentions to invert incorrectly added conditions.

Finally, we added a constraint to scope the fact properties in predicates to the Fact chosen in the FactSelector. This scope constraint made it much easier to select properties in the predicates as indicated in figure 3.5.

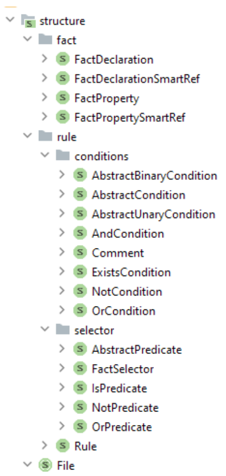


FIGURE 3.2: RSR

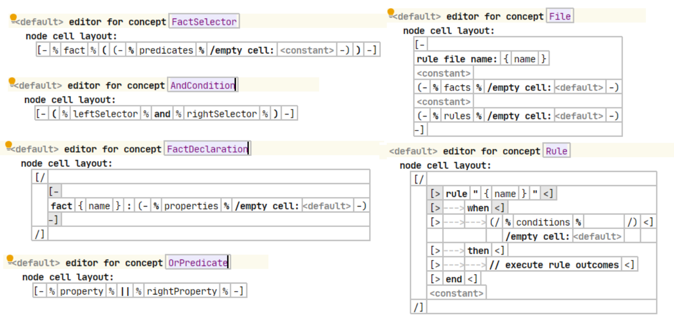


FIGURE 3.3: Editors

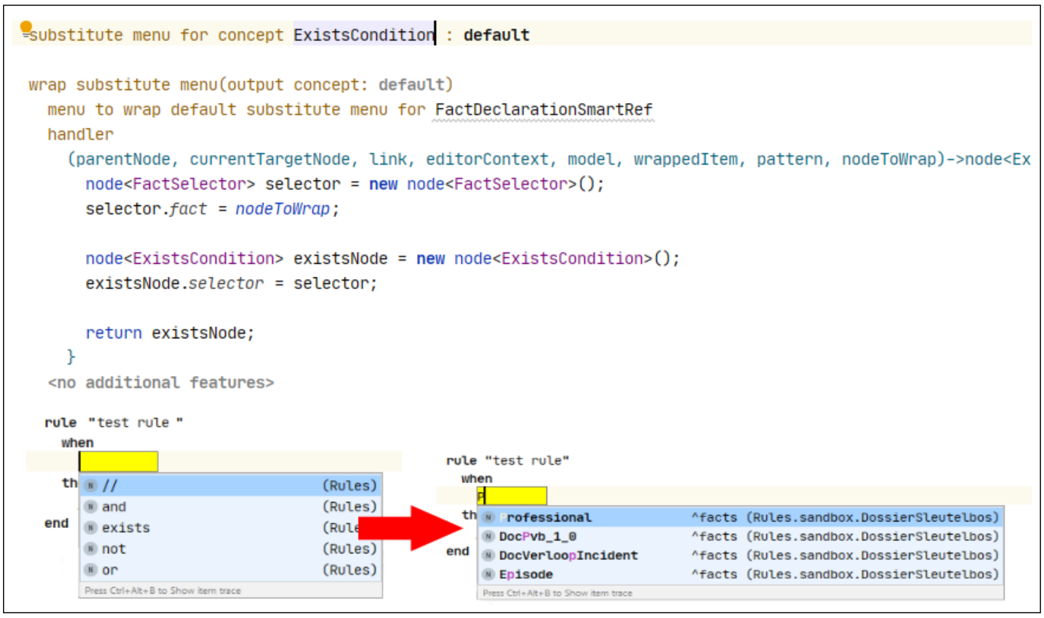


FIGURE 3.4: RSR substitute menu

The figure shows that before adding the scoping constraint, it showed a list with dozens of potential FactProperties, that represented all the FactProperties in the Model. After adding the constraint, it only shows the two properties associates with the Fact from the FactSelector.

Thus, we have described the entire implementation of the Really Simple Rules Language.

After implementing the language, we wrote a program with a large number of rules. This program on which we will experiment with the different projections. Figure 3.6 shows an example of our default Drools like text projection.

We discuss the alternative projections in the results section 4.2.

FIGURE 3.5: RSR scoping constraint

3.2.2 Drools-Lite Language

The RSR was useful as an initial language. However, it suffered from two significant issues. Firstly, its limitations as a language were so substantial that it could not handle many necessary scenarios. Secondly, our projections would have to be validated by developers with Drools experience. For this reason, we needed to create a projectional language that was much closer to the Drools language.

Our following Language, Drools-Lite, contains many more of the features of Drools. Our method of selecting the features involved implementing the examples delivered with Drools (including the corrupt politician example shown in section 2.1.2). We would implement just enough features to complete the examples. Whenever we had any queries about how to design the Concepts, we referred to our analysis of the Drools Language, shown in appendix F. We show the preliminary design we achieved using this method in figure 3.7. Later, there were some places we diverged a little from our design. We merged or decoupled our Concepts when we thought it would simplify the code.

RuleFile The RuleFile level statements contain Facts, Globals and Rules. It also contains semantically unimportant empty lines.

Fact A Fact has a type property. We implement the type property using a ClassifierType from the MPS BaseLanguage. This implementation allows the file to refer to BaseLanguage classes implemented in the same solution or to java JAR files. We created a smart reference Concept for this to take advantage of built-in MPS UI functionality. A smart reference is a node with a single reference of 1:1 cardinality. The editor builders know how to select which nodes in scope to display to the developer if one uses this object rather than directly referencing the node to which it refers.

FactProperty In RSR, we had FactProperties as children of Facts. Now that our Facts refer to actual classes (ClassifierType), our FactProperties should reflect this. To do this, the Concept itself only references an InstanceMethodDeclaration, the MPS BaseLanguage's definition of a Method signature. We scoped the Concept to only show properties associated with a selected Fact.

```
rule file name: DossierSleutelbos

fact Doss ier : hasRunningEpisode , isWvGGZ
fact DroolsCo ntext : << ... >>
fact Episode : isCM , done ,
fact Milestone : cmDecisionStep , finished ,

rule "0"
  when
    Dossier (  )
    DroolsContext (  )
  then
    // nothing
end

rule "[WVGGZ/CM] Start CM procedure"  "
  when
    Dossier ( isWvGGZ , !hasRunningEpisode  )
  then
    // nothing
end

rule "[WVGGZ/VCM] dossier_Start_VCM  "
  when
    ( Dossier ( isWvGGZ , !hasRunningEpisode  ) or ( Episode ( isCM , !done ) and
        Milestone ( cmDecisionStep , finished ) ) )
  then
    // nothing
end
```

FIGURE 3.6: RSR program

Drools interacts with Java objects as if they are Java Beans. To simulate this, we limited the scope of the properties to just getters, i.e. methods that start with "get" or "is" and used a Behavior to make sure they displayed without the "get" or "is" prefix. We also made a smart reference for this Concept.

Another option for achieving this is to have wrapped the ClassifierType and referenced its related InstanceMethodDeclarations. We would have then had to limit the functionality of these items from the BaseLanguage. Whilst this allows the functionality we wished for, we feel our construction offers decoupling and that we think correctly reflects the structure of the language. Perhaps if we were to redo this, we would have taken the other approach.

Global Our Globals are very simple. They have a name and a BaseLanguage Type. We added a smart reference so that Rules can easily use them. The reference extended Expression from the BaseLanguage. This extension is so that we could use it in the Right-hand side.

Rule Our Rules have three children: an Attribute collection, a Right Hand Side and a list of Conditions that make up the Left-hand side. We created a component to describe the rule editor for reuse, as we imagined that we would wrap this in other projections.

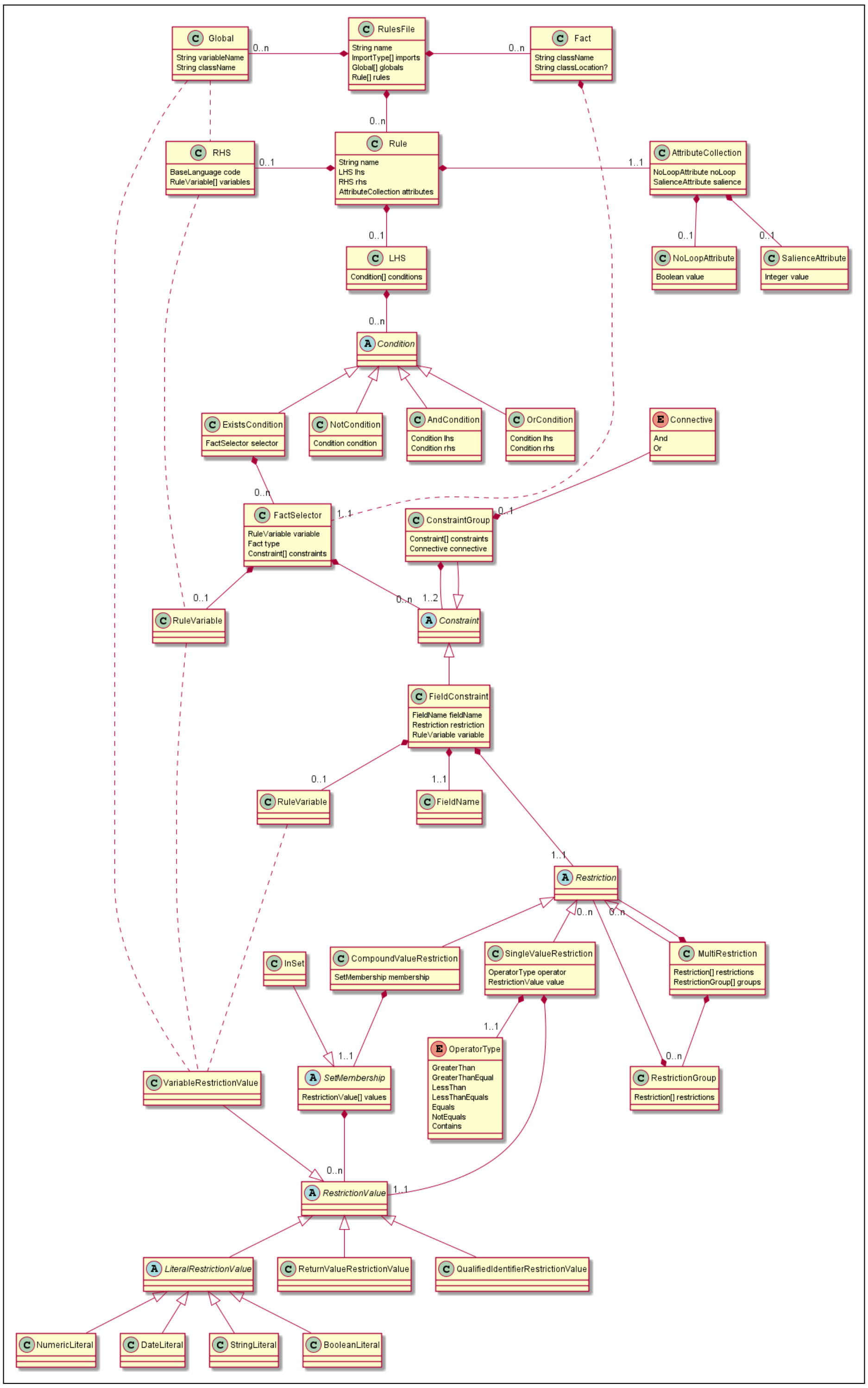


FIGURE 3.7: Drools-Lite structure

RuleVariables The Fact of a FactSelector and the FactProperty of a FieldConstraint are bound to RuleVariables. RuleVariables are scoped to a Rule. A RuleVariable has only a name and a type. We also create a smart reference for it so that it can be used elsewhere within the rule. Like the Global, it extends BaseLanguage's Expression to be available in the Java code of the Right-hand side.

Right-Hand Side The right hand side of the rule is for the most part is Java code. To implement this, we made the right hand side of the rule a single StatementList. A StatementList, from the BaseLanguage, as a list of Statements, also from the BaseLanguage, that keeps track of, amongst other things, scope.

There are a number of non-java, Drools specific items that have to go in the right hand side. Items that had to useable within the righthand side were global variables, rule variables and Drools specific functions. These all extend Expression, so that they can be seamlessly used.

The Drools specific Methods that are required are Insert, InsertLogical, Modify, Delete and Halt.

```
rule "set up"
  when
    $s : Student(  )
  then
    modify( $s ) { setCumlaude( false ) };
    Program program = $s.getProgram();
    insert(  program ) ;

    foreach course in program.getCourses() {
      insert(  course ) ;
    }
end
```

FIGURE 3.8: RHS

Figure 3.8 shows some of the features discussed for the right hand side as shown in our default projection. The right hand side is represented by the text shown between the `then` keyword and the `end` keyword. In the figure one can see examples of plain Java code, such as assigning to the variable `program` and the `foreach` loop. We can also see that Drools-Lite rule variable `$s` is inserted into the Java statements. We have also highlighted the drools specific methods placed in the code, in this case **`modify`** and **`insert`**

RuleAttributes Rule Attributes is a container to hold all of the attributes that are applicable to a rule. Initially, we have only implemented the No-Loop and Salience attributes. These attributes can only be activated through two intentions we added to the Rule Concept. We can see, on line 2 in figure 3.9, an example of the salience attribute added to a rule on line 2. [1]

[1] In 3.9, we added line numbers to this figure to make it easier to talk about. The keywords `rule` on line 1, `when` on line 3, `then` on line 6, and `end` on line 8 have no meaning in the abstract syntax. They are added to give the developer the same look and feel as a standard Drools file.

```
1 rule "Rule #1"
2   salience 10
3   when
4     Program( faculty in ( Faculty.Law, Faculty.FNWI ) )
5     $s : Student( avg >= 8 ) || not Result( grade <= 7 && > 8 )
6   then
7     modify( $s ) { setCumlaude( true ) };
8 end
```

FIGURE 3.9: Rule

Left Hand Side This is a collection of conditions. There are four types of conditions. And, Or, Not and Exists. And, OR, and Not have one or two children who are also conditions. Exists contains a FactSelector.

We added dynamic braces, to only show braces around a condition if it is a child of another condition. This adds visual clarity, without adding unnecessary clutter. we also added some intentions to make it easy to switch between Exists and Not conditions.

On line 4 in figure 3.9, the whole line represents an Exists Condition. Line 5 shows an OR condition with containing an Exists and a Not Condition. The default editor, through an intention, can make the Exists condition explicit with an `exists` keyword. However, the standard practice with Drools developers is to make this implicit, so this is how we show it here.

FactSelector This always has a reference to a fact. These facts are `Program` in line 4 of figure 3.9, and `Student` and `Result` from line 5.

Optionally, the FactSelector can be bound to a variable. In figure 3.9 line 5 the FactSelector referencing the `Student` Fact is bound to the `$s` variable.

The FactSelector also contains a list of constraints on FactProperties, that all must return true in order for the FactSelector to return true.

Constraints We have three types of constraints. And and Or constraints contain other constraints. The FieldConstraints places restrictions on FactProperties.

FieldConstraints A FieldConstraint refers to a FactProperty and can be bound to a variable. It also has a restriction applied to that FactProperty. Using a substitute menu we wrapped the FactProperty smart reference. This automatically creates the FieldConstraint from a FactProperty selection by the developer.

There are several types of Restriction and several types of values that they can restrict.

RestrictionValues Restriction values that a property can be compared with are as follows. Literal Restrictions: These are Integer, Float, String, DateTime and Boolean. Variable Restrictions: these can be global variables, rule variables referring to facts, from the FactSelector, or variables from other FieldConstraints. Return Value: this is for comparing to anything that can be expressed as an expression, which includes referring to constants or values behind qualified identifiers.

In figure 3.9 on line 4 we have the return values `Faculty.Law` and `Faculty.FNWI`. on line 5 the literal values `7` and `8`.

Restrictions A SingleValueRestriction compares a FactProperty against a value. A MultiRestriction compares a FactProperty against multiple values, not necessarily using the same comparison for each value. A SetMembership restriction checks if a FactProperty is a member of a group.

In figure 3.9 on line 4 a SetMembership restriction is shown with the `in ( Faculty.Law, Faculty.FNWI )` text. Line 5 in the first FactSelector there is the SingleValue restriction `avg >= 8`. The second FactSelector shows a MultiRestrictions `grade <= 7 && > 8`.

Thus, we have described the pertinent implementation details of the Drools-Lite language.

3.2.3 Wireframes

There are some potential projections we have conceived for which there is not sufficient time to implement. We would like for these to be assessed and thus would like them to appear as real as possible to the assessors.

Our solution to this conundrum, is to develop these presentations in a wireframing tool. The Wireframe tool we choose was Axure[74]. We chose thus as we had previous experience of the product. Also, it is available to students for free.

After much discussion, we settled on two possible projectional programming aids: Truth table and circuit diagram. We will discuss these in more details in the results section.

3.3 Method: Survey

The validity of the the Prototype was tested using a survey. If a survey is not well designed then it could lead to invalid or irrelevant outcomes. As well as describing the design and procedure of the survey, we also outline any threats to its validity in this chapter. Our choice of survey technique is a Questionnaire.

3.3.1 Questionnaire Design

The questionnaire can be found in appendix G. To design the survey of our prototype, we followed a number of rules derived from the works of Bryman[75] and de Vaus[76].

As advised, first we devised a clear introduction to describe the research.

We considered existing questions. With regards to projectional editing, we requested the original questionnaires from three papers[30, 35, 53] pertaining to tools developed using projectional editing. From these questionnaires we found [X] that we considered and decided against using any of them.

When formulating the questions we had the specific research question “Which projections can help developers to get appropriate feedback about rules?” in mind.

The pool of Drools users that we were personally in contact with was incredibly small. Thus we had to rely on responses from strangers. For this reason, we tried to make the questionnaire as quick to finish as possible. This meant we looked particularly hard at removing questions that did not help us to our research goal.

We piloted the questionnaire with both ourselves and our industrial supervisor.

The instructions to each of the questions were tested for clarity, by a non-technical third party.

The only open question was one for which we wished to extract sentiment. Rather than having yes/no questions, where appropriate we applied a Likert scale[77].

The design of survey monkey layout makes sure that questions do not span pages.

The socio-demographic questions (skill level and experience) were left to the end and research based questions were toward at the beginning.

We took care to rework questions that were long, ambiguous, general, or leading to not be so. We also took care to remove jargon, negative wording, and questions that asked about more than one thing.

3.3.2 Participants

The requirement for participants is that they have at least a little experience with using Drools. It was our hope to get a statistically significant number of participants.

3.3.3 Validity

Non-response bias[78] will be addressed by making the questionnaire short and easy to answer. Because of the nature of the participant selection for this survey, it will be difficult to address the bias of self selection caused by voluntary response.

Common method bias, i.e. “variance that is attributable to the measurement method rather than to the construct the measures represent”[79] can be responsible for 25% or more of variable relational influence. As we are only conducting a single survey, we won’t be able to do much to prevent this, however we will take the following small precautions. Testing the survey to remove question ambiguity, mood influences, and length issues. Mixing the survey order of questions will be used to mitigate the issues caused by similarity of items, proximity of items, and location of items. We will mitigate survey administration biases by administering some of these questionnaires manually and some online. We will make sure that there are no right or wrong answers and aim toward fact based questions. We will vary the scales of our Likert scales and the types of questions.

The main statistical methods to address this bias, i.e. “Harman’s single factor test”[79] and the“marker variable”[80] were found to be lacking in grounding[81]. Marker variable is considered appropriate if used with caution. With the size of our expected returns, it may not be possible to gain a statistically significant outcome.

3.3.4 Pre-test

The first attempt at the survey was sent to our industrial supervisor, who has experience with Drools. We used this to remove ambiguously worded or leading questions and test that the length was truly between 5-10 minutes. This lead to the following changes: [TODO: add changes after we have tested]

3.3.5 Sampling

As within our own professional network we only had acquaintance with (6?) Drools developers, we had to expand our reach to those we did not personally know.

Our approach was first search StackOverflow for question askers and answerers on the subject of Drools. Our preference was to find email addresses, failing that twitter contacts. This proved to be quite limited, especially in attempting to get contact details, 13 email addresses and 6 twitter addresses.

Our next approach was to trawl our LinkedIn connections for anyone with drools as a proclaimed skill. Whilst we had no one in our direct contacts, at one level of separation we found 204 connections. From these we could extract 54 email addresses and 40 twitter addresses, with only a small crossover with the addresses harvested from StackOverflow.

We chose not to expand to third level contacts, as we thought this would be harder to sell as to why they should feel comfortable answering us.

3.3.6 Procedure

The questions, as described in appendix G, was uploaded to Survey monkey.

To encourage response, especially amongst only tangentially known participants, we crafted a short introduction, using techniques designed to enhance response as discussed by amongst others Cialdini[82]. As seen in figure 3.10, the attics discussed by Cialdini, as signposted in table 3.4.

Key	Tactic
1	Short option for those with no time
2	Credentials matter
3	Recognition, (this might backfire as I hardly remember any of my LinkedIn connections)
4	Consistency, they reported they have Drools experience, so they must live up to it
5a&b	Social Proof - other people have already answered
6	showing value
7	Special because of scarcity
8	Labelling - “I see you to be a good person”
9	The word “Because” has an outweighed effect
10	Compliment Expertise
11	“Every little helps”
12	point out a fault
13	own the fault
14	ask a favour
15	add inconvenience
16	Rhyming
17	hand written note

TABLE 3.4: Persuasion tactics in figure 3.10

These were sent to our list of drools using strangers and we sat back and awaited response.

1 TL;DR 5 Minute Drools Survey for Masters Thesis [URL here]

Dear (NAME),
I am Paul Spencer, I am completing my Masters Degree at the University of Amsterdam. 2
My final requirement, my thesis, requires that I conduct a scientific enquiry involving Drools users of various experience levels [URL].
I am contacting you because you are connected to my contact 3 (name) in my LinkedIn network and because you report you have experience with Drools. 4

I am currently very close to reaching statistical significance 5a in my survey and your response could get me there. 6
I only have a small pool of connections with Drools experience, however I have achieved an 82% response rate so far. 5b 7

I expect you will want to participate 8 because 9 I see from your profile a person who has dedicated a lot to computer science over the years. 10
However, if you cannot spare the 5-10 minutes to fill the questionnaire, please answer one or two as each answer helps. 11

13 12
I am embarrassed to say this, but, through my own negligence, I left this email survey a little late, so if you are able to do me the favour of answering this as soon as possible, I may be able to graduate this academic year rather than next! 14 15

16
In summation, please answer this survey of a Drools Rule tool, so this fool can leave school!
[URL]

thank you 17

FIGURE 3.10: Persuasive introduction

Chapter 4

Results

> the purpose of abstraction is not to be vague but to create a new semantic level in which one can be absolutly precise.
>
> *The Humble Programmer*
> *Edsger W. Dijkstra*

4.1 Results: Systematic Literature Review

Using a systematic literature review (SLR), following the method described, to answer the question, "What is the current state of Projectional Editing", leads us to the conclusion of - undetermined.

We make this statement because the review design was not appropriate for the problem domain. The SLR was abandoned after the quality assessment stage. We were unable to find a quality assessment checklist that adequately accounted for action design research, which most of the primary studies were. There were some self-identified case studies which in fact were only descriptions of usage of implementations, rather than case studies in a scientific sense.

Therefore what follows should be considered as the results of a Quasi-SLR.

4.1.1 Papers Selected

The details of what is described in this section are logged in appendix B

Figure 4.1 shows the results of the 5 iterations that the search went through. From our initial search using the scholarly search tools out of 173 results we had 50 papers that initially seemed to pass our inclusion and exclusion criteria. From the initial 50, we added 18 papers from a possible 109 in our first iteration of forward and backward snowballing. The next snowballing iteration returned three papers that matched our criteria. The third round of snowballing had no matching papers and thus terminated this stage of selection.

Our final selection iteration involved a deeper scan of to determine of the remaining 71 papers. With this we could reject 12 papers which were not primary research, one paper

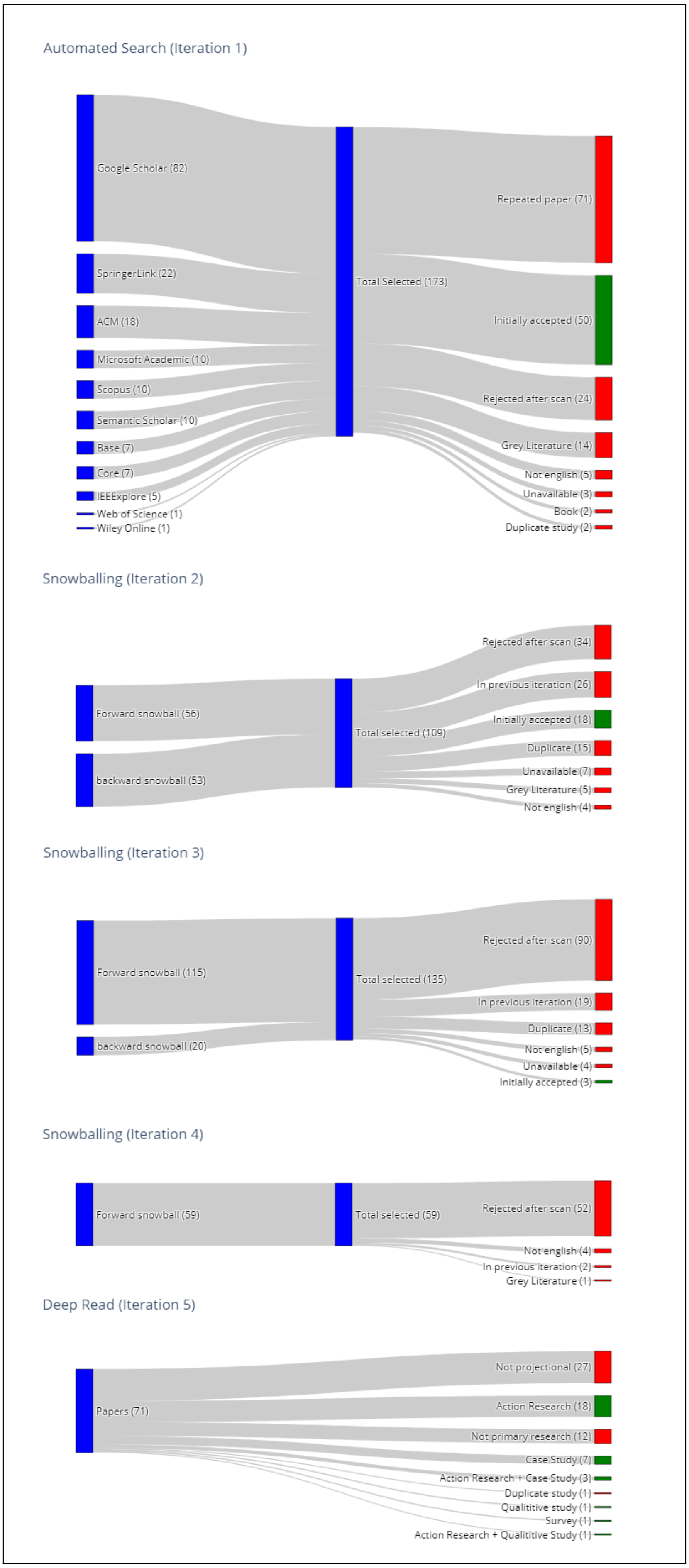


FIGURE 4.1: Search results

that reported a study already represented. There were also 27 papers that were, on closer reading, not about projectional editing.

This left us with 31 papers before the quality assessment filter.

Sensitivity and Precision

As a curio, we reapproriated Zhang's[65] ideas of sensitivity and precision and applied them to the search engines rather than search strings. The values for sensitivity and precision of the search engines are calculated as follows:

$$sensitivity = \frac{\#\, retrieved\ relevant\ studies}{all\ relevant\ studies} 100\%$$

$$precision = \frac{\#\, retrieved\ relevant\ studies}{\#\, studies\ retrieved} 100\%$$

Table 4.1 show that Google Scholar had the highest sensitivity, returning 22 of the 31 chosen studies. This came at the cost of a very large proportion of false positives. Microsoft Academic and SpringerLink were joint most precise with half of their search results ending up in the final roster. SpringerLink, with the second highest count of documents, second highest sensitivity, and joint highest precision would appear to be the best all around search engine for this field. However, these figures are skewed by several of their articles coming from a single collection specifically about projectional editing.

Search engine/library	original #	selected #	sensitivity	precision
ACM	18	3	10%	16%
BASE	7	3	10%	43%
CORE	7	1	3%	14%
Google Scholar	82	22	71%	27%
IEEExplores	5	2	6%	40%
Microsoft Academic	10	5	16%	50%
Science.gov	0	0	0%	0%
SCOPUS	10	3	10%	30%
Semantic Scholar	10	4	13%	40%
SpringerLink	22	11	35%	50%
Wiley Online	1	0	0%	0%
Web of Science	1	0	0%	0%

TABLE 4.1: Search engine sensitivity and precision

4.1.2 Quality Assessment

Using the quality assessment checklists, developed by Crombie et al.[83], shown in appendix C we examined the remaining 31 papers which on the surface represented 37 primary studies. As action design research was not represented amongst these checklists we searched for an appropriate quality assessment checklist for these type of primary study. We did not find a suitable checklist, and did not consider ourselves suitably qualified to make one. Thus we used the case study quality assessment checklist to assess the ADR studies.

We used a rudimentary scoring system of +1 value for positive answers, 0 for “don’t knows”, and -1 for negative answers. We arbitrarily chose that any study with an overall score greater than 0 would be considered high enough quality to be part of our final analysis.

With this scoring and goal we only found 6 out of 37 studies of high enough quality.

We therefore had to decide to change our scoring, go ahead with 6 studies, stop here or ignore the QA findings. Changing the scoring until you get the “right” result felt wrong. 6 studies seems too few to give an overview of a field. Stopping seems to be the correct course for an SLA. However, we decided to create a new Quasi-SLA, that ignores the results of a Quality assessment.

Our reason to continue was that we could not reconcile that 84% of studies that had made it to the recognized scientific journals were not of high enough quality to pass our SLR QA stage. We believe there were two large threats to the validity of the Quality Assessment stage that were too great to ignore. The first being that it was carried out by a single researcher with no previous experience. The other being either that the use of case study checklists is inappropriate for ADR studies or that ADR studies are inappropriate for SLRs.

4.1.3 Analysis

After Identifying the primary studies we extracted data, the details of which are shown in appendix E. Figure 4.2 shows that most of the primary studies in our review were ADR studies.

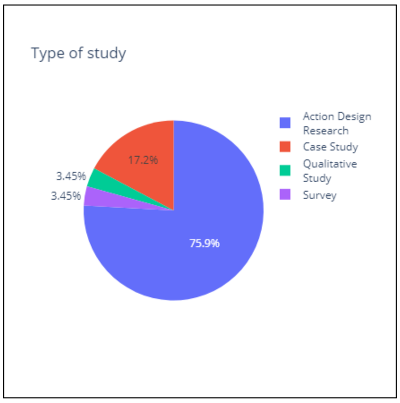


FIGURE 4.2: Study types

Tools used

We split the studies to see which were to do with purely research projects and which were researching using already publicly available commercial or opensource products. To calculate this we removed the one Primary study that was a survey, as it covered a multitude of tools and options, none of which in depth. The chart in figure 4.3 shows that over 80% of the projects were studying already existing publicly available options.

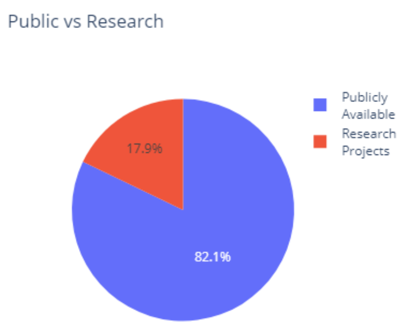


FIGURE 4.3: Public vs research

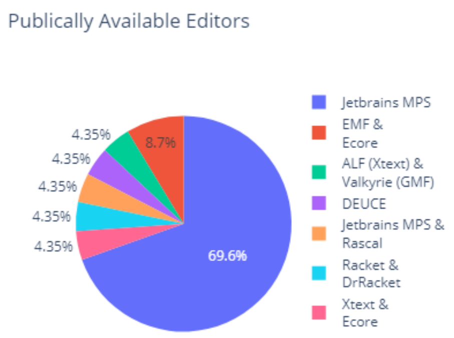


FIGURE 4.4: Publicly available programs

Of the studies that using on publicly available software, we wanted to know what software was attracting most academic interest. Figure 4.4 shows that 74% of the studies into projectional editing that were using a publicly available product were conducted using Jetbrains.

Sentiment

In this study we included all 29 papers. We tagged each section from those papers that talked about projectional editing. We then broke each of these sections into sentences and ran those sentences through a sentiment analyser as described in section 3.1.6. The outcome of these are shown in figure 4.6.

The charts show the relationship between the positive, neutral and negative sentiment outcomes. On the y-axis, we have an Id for the papers being examined. A key linking the Id to the paper name is seen in table 4.2;

The first chart shows the absolute number of sentences that were analysed per paper partitioned by whether they returned negative, neutral and positive. The second chart shows these as percentages, so that the papers are more comparable. In the third chart we removed the neutral scores and calculated the percentage positive to negative. The final chart is an aid to make it easier to scan whether papers were more positive, more negative. If we found the papers to be equally positive and negative, we classified them as neutral.

Over the 29 papers we scanned a total of 3003 sentences. 435 were analysed as being positive, 1953 neutral and 615 negative. Thus 14% were positive and 16% negative.

10 of the 29 papers were more positive than negative when talking about projectional editing, 16 more negative and 3 equally negative and positive.

In figure 4.5 we attempt to separate out sentiment by product category. These categories being Research projects, MPS and all the other used products. We ignored the survey paper in this one as it covered all of these types.

As MPS dominates the sentences accounting for 17 (61%) of the 28 papers and 2051 (73%) of the 2791 sentences analysed, the comparison does not give much value.

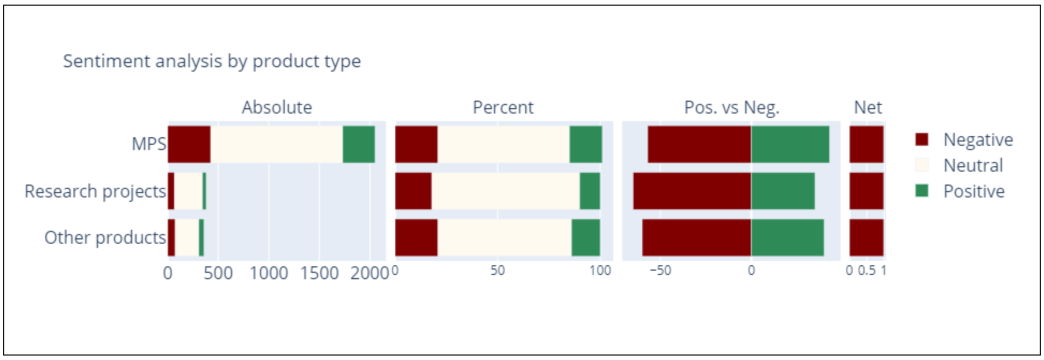


FIGURE 4.5: Sentiment analysis by product

A Narrative Synthesis

Our synthesis of the papers that appear in table 4.2 will be short. We will avoid rehashing the advantages and disadvantages of projectional editing, as discussed in sections 2.2.3 and 2.2.3.

There is much focus in these papers on models, and model driven development. Some mention is made of a shift towards textual modelling languages. However, others point out that the suitable level of abstraction is not always found in text.

When authors have used solutions other than MPS, they complain about issues such as MPS being heavy-weight, with a lot of overhead. However, these authors then spend a great deal of time theorising about how to fix issues that are fixed by default in the implementation of MPS. The projects they describe are not in industrial use. These authors think that projectional editing is probably best suited to helping novices learn a language.

There is a fair bit of mention of "semi-projectional" approach, where there is parsing at the leaf level. This is mainly from those who don't use MPS, but also some who do. This, it seems, is a reaction to the difficulty in simulating the text language experience in a projectional editor.

Those authors who use MPS are mostly either discussing products developed for use in an industrial setting or discussing how best to teach projectional editing. When MPS is used it is often seen as a key enabler.

The

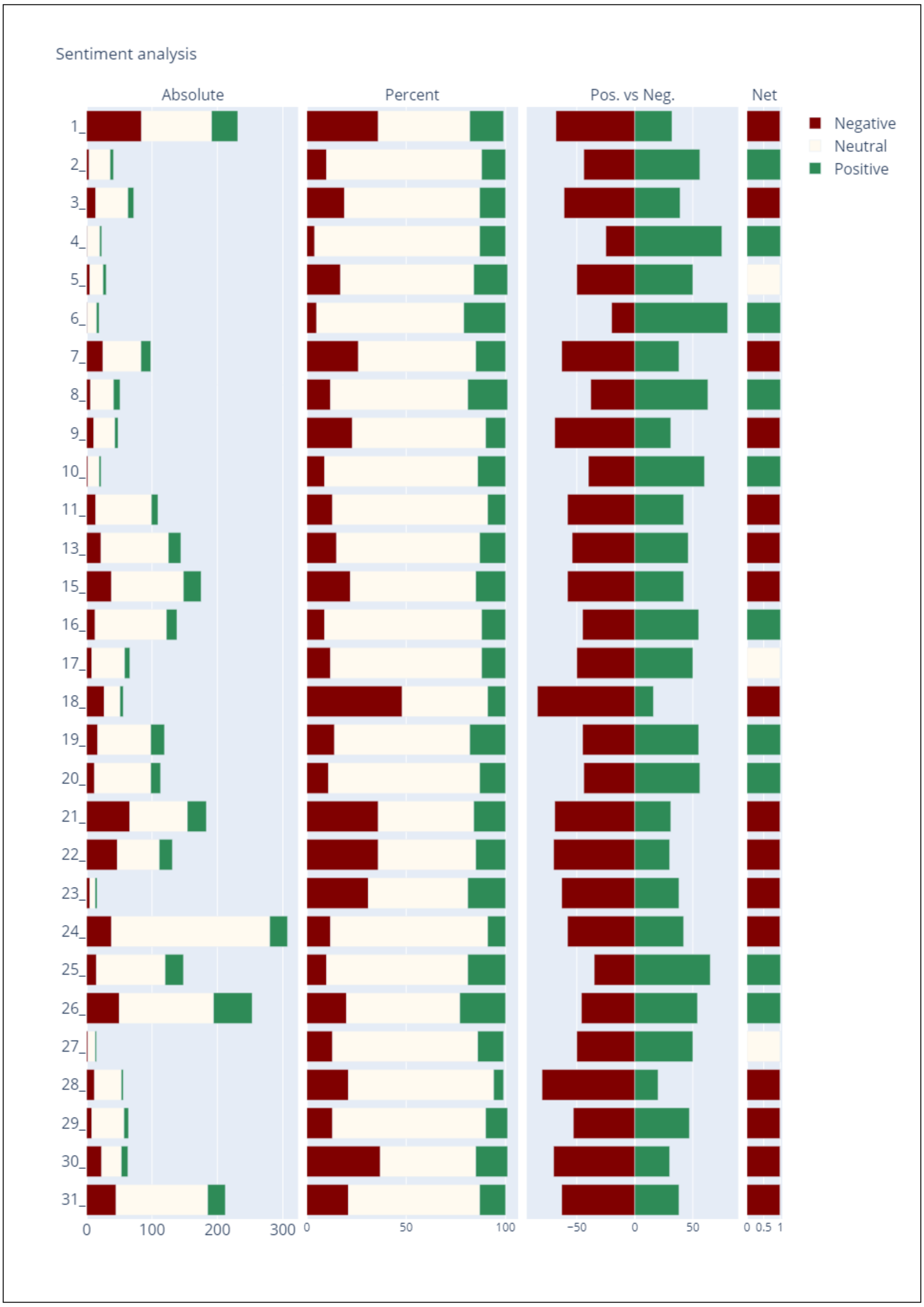


FIGURE 4.6: Sentiment analysis

Id		Paper name
1	[84]	A domain-specific language for payroll calculations: A case study at DATEV
2	[85]	A framework for projectional multi-variant model editors
3	[86]	A generic projectional editor for EMF models
4	[87]	A model-driven approach towards automatic migration to microservices
5	[88]	AdaptiveVLE: An integrated framework for personalized online education using MPS JetBrains domain-specific modeling environment
6	[89]	Adding interactive visual syntax to textual code
7	[90]	Blended graphical and textual modelling for UML profiles: A proof-of-concept implementation and experiment
8	[91]	Classification algorithms framework (CAF) to enable intelligent systems using JetBrains MPS domain-specific languages environment
9	[92]	DSL based approach for building model-driven questionnaires
10	[93]	Efficient editing in a tree-oriented projectional editor
11	[94]	Efficient generation of graphical modelviews via lazy model-to-text transformation
13	[95]	Engineering gameful applications with MPS
15	[96]	Fasten: An extensible platform to experiment with rigorous modeling of safety-critical systems
16	[25]	Gentleman: A light-weight web-based projectional editor generator
17	[97]	Integrating UML and ALF: An approach to overcome the code generation dilemma in model-driven software engineering
18	[98]	Javardise: A structured code editor for programming pedagogy in Java
19	[99]	Jetbrains MPS as core DSL technology for developing professional digital printers
20	[100]	Learning data analysis with metaR
21	[101]	Migrating insurance calculation rule descriptions from Word to MPS
22	[102]	Model-based safety assessment with sysml and component fault trees: Application and lessons learned
23	[103]	Papyrus for gamers, let's play modeling
24	[104]	Projecting textual languages
25	[105]	SpecEdit: Projectional editing for TLA+ specifications
26	[106]	Teaching language engineering using MPS
27	[107]	Teaching MPS: Experiences from industry and academia
28	[108]	Tiny structure editors for low, low prices! (generating guis from toString functions)
29	[109]	Towards ontology-based domain specific language for internet of things
30	[110]	Type-directed program transformations for the working functional programmer
31	[111]	What do practitioners expect from the meta-modeling tools? a survey

TABLE 4.2: paper key

4.2 Results: Action Design Research

4.2.1 Really Simple Rules

The Really Simple Rules, (RSR), acted as a training ground for our new projections.

Context Aware Color Scheme

After the default text projection, the first projection we made was giving the text a color scheme. This form or augmentation in IDEs is probably the most basic that we see. Available in structured editors since the 1980s[112], syntax highlighting displays text in various colors and fonts according to the meaning of the terms. Syntax highlighting has been found to be useful for comprehension of code, as least for small code bases[113].

Developers at our host organization uses Eclipse or IntelliJ Community Editions to edit code, neither of which have syntax highlighting for Drools, thus the addition of this feature would immediately benefit them. However, IntelliJ IDEA, already provides this feature for Drools. In order to offer another visual augmentation that we considered should be useful we extended the color scheme to indicate whether the selection is looking for a positive or negative match. This is shown in figure 4.7.

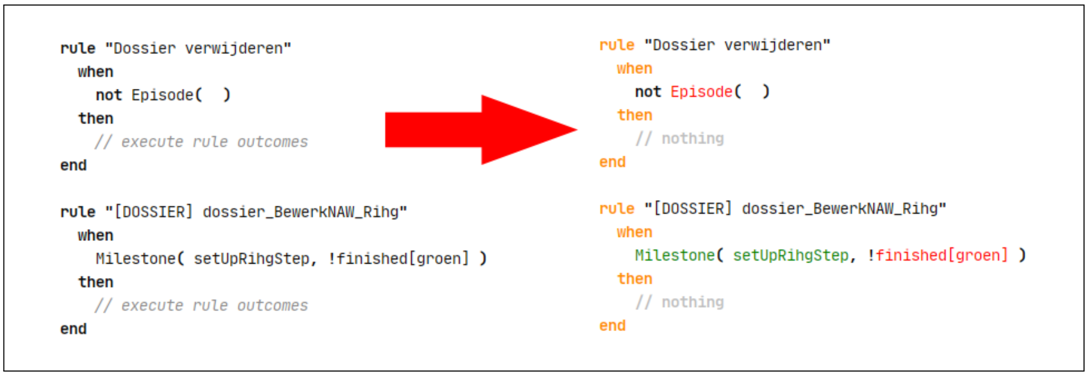


FIGURE 4.7: Context aware color scheme

Facts that are contained by NotConditions and FactProperties that are part of a NotPredicate are highlighted in Red. ExistConditions and IsPredicates have their content colored green. We did not test whether this improved understanding.

Summary Projection

Our next projection allows developers to have a quick overview of rules and complexity of those rules. Figure 4.8 shows that the developers are able to get an overview of both the number of rules and the number of facts in each of the rules.

The building of this rule only required adjusting two editors. The rule count and fact count were added to the File editor using Read Only Model Access, to just count the descendants of the file that are rules and facts. The rule editor was adjusted to only show the title of the rule and, again using the model access, the count of the descendants of the rule that were facts.

```
rule file name: DossierSleutelbos  rule count 152  fact count 357

0 : # facts:  2
Dossierdetails kunnen inzien : # facts:  1
Dossiers samenvoegen : # facts:  0
[*] dossier_NieuweNotitie : # facts:  1
[*] dossier_StuurVeiligeMail : # facts:  1
Dossier verwijderen : # facts:  1
[HV ] dossier_Sluiten : # facts:  1
[HV ] dossier_Afdoen : # facts:  1
[HV ] dossier_Actualiseren : # facts:  1
[WVGGZ/VO] Bewerk NAW in WVGGZ VO voor meldmedewerkers : # facts:  3
[WVGGZ/IBS_WZD] Bewerk NAW in WVGGZ IBS_WZD : # facts:  1
[WVGGZ/CM] Start CM procedure" : # facts:  1
[WVGGZ/CM] Verzamel Stuurgegevens : # facts:  2
[WVGGZ/CM] Wijzig Stuurgegevens Voor Afronden Medische Verklaring : # facts:  3
[WVGGZ/CM] dossier_Act_Opstellen_CMMedischeVerklaring : # facts:  2
[WVGGZ/CM] dossier_Act_Onderteken_MedischeVerklaring : # facts:  3
[WVGGZ/CM] dossier_TerugzettenAfgerondeCMMedischeVerklaring : # facts:  5
```

FIGURE 4.8: Summary projection

Whilst, this may look like a report, that any language workbench could create, the file name and all of the names of the rules are, in fact, still editable in this projection.

Filtering

The nature of business rules lends them to some projectional options that would not make sense with other programming styles. because of the small, independent nature of the rules filtering in particular lends itself to the business rules style. When looking at how to understand large files, we sought other domains than programming that handle large volumes of items. The domain of data has a long history of handling large volumes. Amongst their two most used tools for exploration are Sorting and Filtering. On top of the semantic meaning of the ordering of the rules, we also did not imagine a good use case for sorting rules. Thus de decided to implement a filtering projection.

Whilst filtering has been used in other places in the coding pipeline, such as in deciding on what code completion to present[114], and version control visualization[115], we were unable to find any research on applying filtering directly to code files. Thus, we think what we present here is an original implementation.

When thinking about how business rules are related, one of the first things we look at is rules that examine the same facts or fact properties. Thus these seemed the most obvious items to filter on. We created a projection where if the developer filtered by a Fact or a Fact Property, then the projection would only filter out all rules that did not contain the item. On top of this once the rules were filtered it would only show Facts and Fact properties that were used by those rules.

In our implementation, shown in figure 4.9, we show three places where we use intentions to filter the code. The first is an intention on a Fact. The outcome of choosing this filter is shown on the righthand side of figure 4.9. The second intention is on a FactProperty. As the FactProperty is a child of a Fact and we enabled that children could also show intentions, we also see an intention to filter on the parent Fact type. The third highlighted intention is on a FactProperty Reference. It also shows an intention for a Fact Reference in the FactSelector that holds the FactProperty reference as a child.

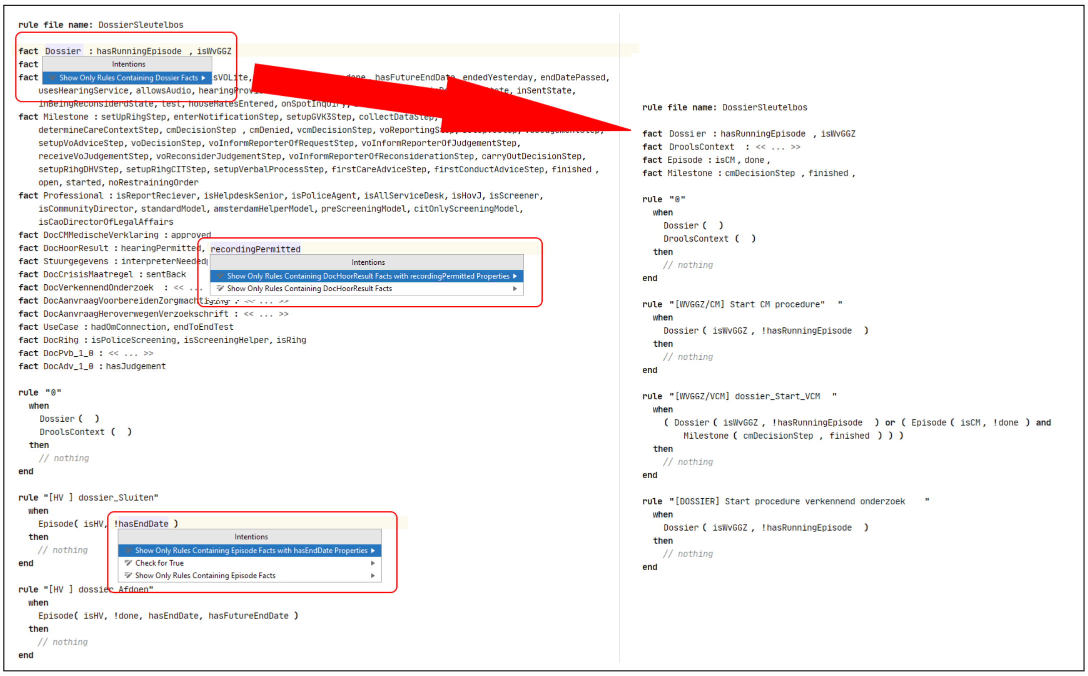


FIGURE 4.9: Filtering projection

One of our guidelines for ourselves was, as much as possible, to build our projections as separate languages, non-invasively extending RSR. Our first approach at the filtering, we failed on this count, by adding properties to Fact and FactProperty Concepts, to determine whether they were visible.

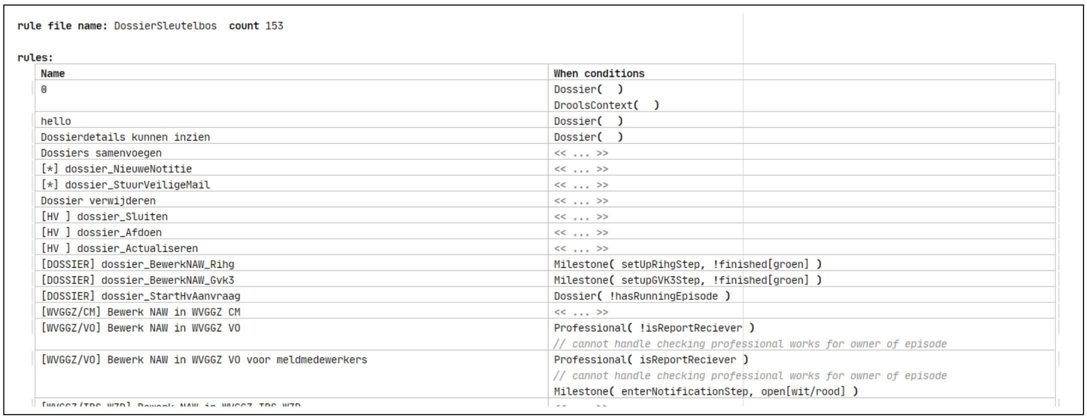
Our next approach created subclasses of Fact, FactProperty and File. This however requires running a macro on the code file to migrate Facts, FactProperties and Files, to FilteredFacts, FilteredFactProperties, and FilteredFiles. This means that the file could only be used by other languages that extend the filtered language.

Our Final approach was to add a Filter Concept and have that reference the filtered nodes, and have the editors make the visibility calculations based on this singleton node. Whilst more complex, this removed the need for invasive changes and allowed other languages to combine with the filtering language.

Whilst this filtering is a handy projection, it does break Dijkstra's rule "the purpose of abstraction is not to be vague but to create a new semantic level in which one can be absolutely precise."[116]. The Summary projection could have been extended, so that the inspector contained all of the details of a selected rule. This projection has no way of containing the whole code whilst a filter is applied. However, so long as there is a clear indication that a filter is applied, then I do not see this as more of a problem than code collapsing found in most modern day editors.

Table

Thus far our projections have been textual ones that can be imagined with other language workbenches. Creating a table will be our first non-parseable projection. Our choice for this projection is based on the of Miller's[2] observations about the number of items people can retain in their memory. Based on this observation then the counter-measure to this is to have fewer items off the screen and therefore in the developers memory.


rule file name: DossierSleutelbos count 153

rules:

Name	When conditions
0	Dossier( ) DroolsContext( )
hello	Dossier( )
Dossierdetails kunnen inzien	Dossier( )
Dossiers samenvoegen	<< ... >>
[*] dossier_NieuweNotitie	<< ... >>
[*] dossier_StuurVeiligeMail	<< ... >>
Dossier verwijderen	<< ... >>
[HV] dossier_Sluiten	<< ... >>
[HV] dossier_Afdoen	<< ... >>
[HV] dossier_Actualiseren	<< ... >>
[DOSSIER] dossier_BewerkNAW_Rihg	Milestone(setUpRihgStep, !finished[groen])
[DOSSIER] dossier_BewerkNAW_Gvk3	Milestone(setupGVK3Step, !finished[groen])
[DOSSIER] dossier_StartHvAanvraag	Dossier(!hasRunningEpisode)
[WVGGZ/CM] Bewerk NAW in WVGGZ CM	<< ... >>
[WVGGZ/VO] Bewerk NAW in WVGGZ VO	Professional(!isReportReciever) // cannot handle checking professional works for owner of episode
[WVGGZ/VO] Bewerk NAW in WVGGZ VO voor meldmedewerkers	Professional(isReportReciever) // cannot handle checking professional works for owner of episode Milestone(enterNotificationStep, open[wit/rood])

FIGURE 4.10: Table projection

Figure 4.10, shows our rudimentary first table. This simple table has only the Name property and the when children of the rules in the file. This is implemented using the table extension, in the MPS-Extension Plugin, created by Sascha Lißon.

Cross-Tab

Our next tabular projection is a cross-tab inspired by a decision table. The idea for this is that the previous table does not give any visual queues as to how rules are related. With a cross-tab one can easily see which rules are using the same Facts.

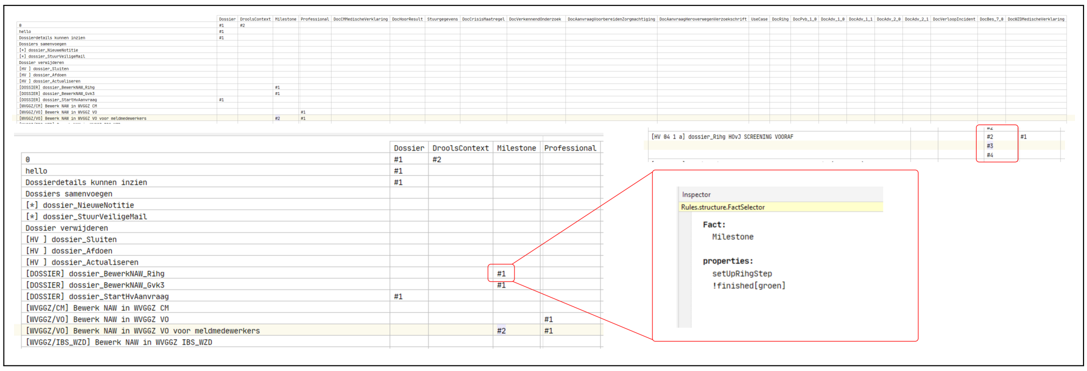


FIGURE 4.11: Cross-tab projection

Figure 4.11 Shows our implementation of the cross tab. At the top we can see an immediate problem with a cross-tab, and that is, if we have the whole file included the table will be very sparse. Figure 4.11 also has a close-up of a cell showing a rule using three FactSelectors that reference the same fact. The other close-up shows that all the details of the selected fact are available in the inspector.

The sparse table would not be a problem if the columns are thin enough to keep the table in a single screen width.

Everything is editable in this table, including deleting the fact from the rule. Most of the editing was provided by default by either MPS or the table plugin. An extra editing feature we added to this table was the ability to delete a Fact from the file, and thus deleting all references to it from all the rules in the file, by deleting a Fact column column. The code shown in figure 4.12, shows how we can walk the trees in each rule to delete unary conditions and convert the non-deleted side of binary conditions into unary conditions.

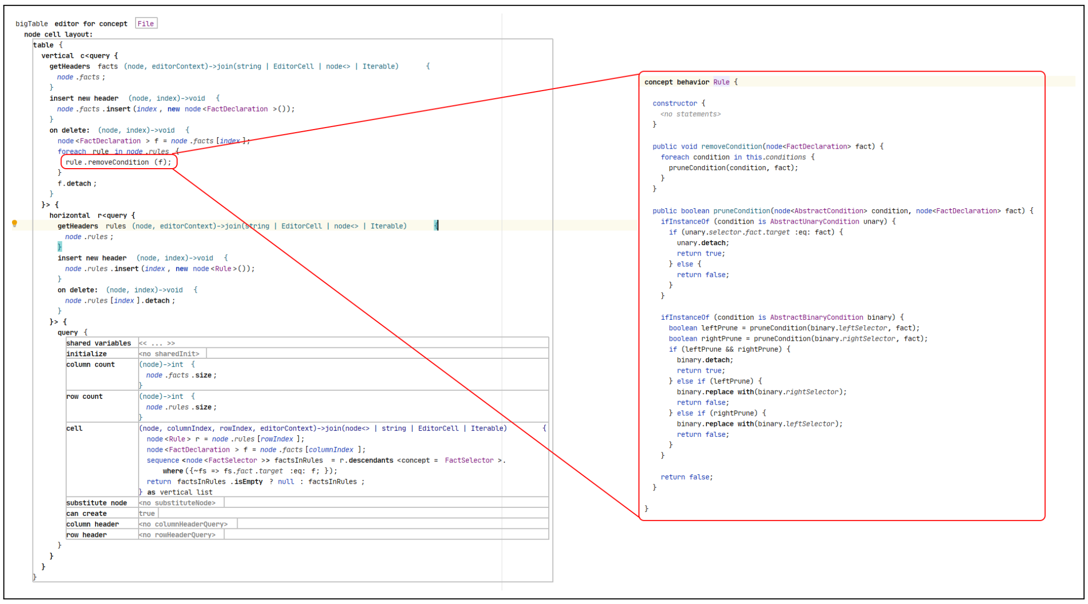


FIGURE 4.12: Table fact deletion code

Here ends our experiments in the RSR language.

4.2.2 Drools-Lite

Our next experiments were with projections with the Drools-Lite language. As described in the section 3.2.2, is an implementation that is much closer to the full Drools language. These are the projections that we will present to experienced Drools developers for evaluation.

Of the learnings from the RSR language, one we felt needed to be fixed to aid understanding was the sparseness of the tables. By implementing the principle of maximising cohesion, we discovered we could reduce the sparseness issue. Thus, as a precursor to our projections, we extended Drools-Lite with a new language that contained one structural item, the Rule Collection. The rule collection sits at the file level and holds a collection of rule statements. The idea behind this is that rules that are related can be placed in the rule group so that it is easier to examine them together. This language also had a default editor, and intentions to move rules in and out of groups.

Decision Table

As the Drools Language is analogous to a series of it-then-else statements, then perhaps it's best visual equivalent is the decision table. It has long been shown that decision tables are a "powerful aid in programming, documentation, and in effective man-to-man and man-to-machine communications"[117].

We designed our table, shown in figure 4.13, to include some of the lessons learned from the RSR CrossTab that was demonstrated in figure 4.11. The RSR language taught us that sparseness in tables is exacerbated through wasting of visual real estate. In the crosstab table, horizontal scrolling was necessary in part due to the column widths. The columns were wide as the name of the Fact was displayed horizontally.

The Drools-Lite language allows for much longer selection criteria on Fact Properties, so we had to design the display with this in mind. This screen real estate needed to be saved, as our decision table required a column to show the consequences of the selections, which in itself would be a wide column. Our solution to this was to develop a vertically orientated header cell and use indentation to indicate if the cell is referring to just the Fact or a Fact and Fact Property combination. The text for the header cells are generated based on the current

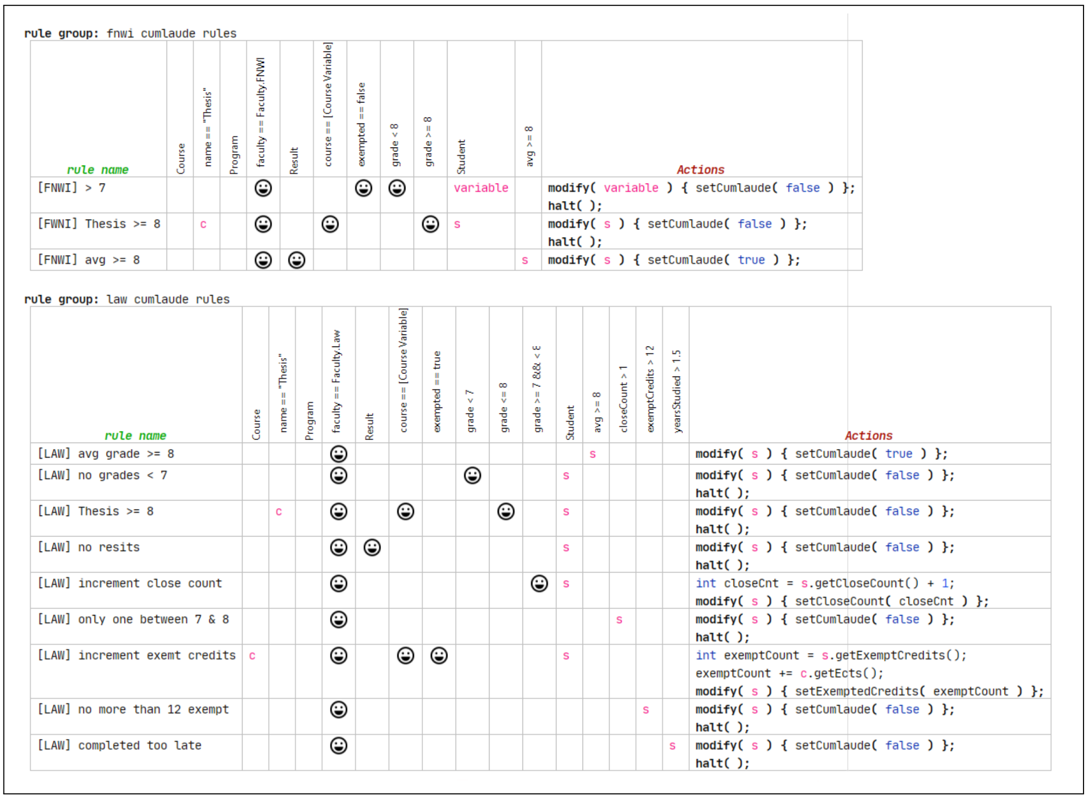


FIGURE 4.13: Decision table projection

Because this projection presents both the left and right hand side of the rules, we had to handle the Concept that spans both of these - the RuleVariable. As the RuleVariable can be bound and used on the LHS as well as being used on the RHS, we had to find a way of representing this. This was achieved by allowing a variable name to be used in the cell that represented the Fact or FactProperty it represented. This lead to a happy visual design choice. With variables now being represented in the cells, we could no longer represent the cell being selected with an 'X', as this could lead to confusion as to the meaning. Projectional editing does not require meaning to be communicated with ASCII text. Thus we decided to represent selection with an image, and personal preference and a misspent youth lead us to the smiley face as that indicator.

The rule names and actions are editable through the default functionality of the MPS extension. Adding a selection criteria to a rule occurs through an intention attached to the associated cell, as is also the case for binding a variable.

The major drawback of this design is that editing a rule with an as yet non-existent selection criteria became very clunky. If the rule we wished to edit already existed in the table we had to use an intention to extract it from the group change the criteria and place it back in. At this point the table would automatically adjust the column headings.

This design was examined by experts in the questionnaire.

SpreadSheet

The domain specific language for the finance world is the spreadsheet. One study estimated that 90% of computers had a spreadsheet on them[118]. It has been suggested that Dan Bricklin's VisiCalc is the reason for the success of PC's in The office. VisiCalc was succeeded by Lotus 1-2-3, which in turn was succeeded by Microsoft Excel as the dominant Spreadsheet program in the workplace.

This level of familiarity to a paradigm lead us to try to design a projection that had the same look and feel of an Excel spreadsheet. This design can be seen in figure 4.14. To this end we created a design where the selection criteria could be directly edited in the cell, as highlighted in the figure.

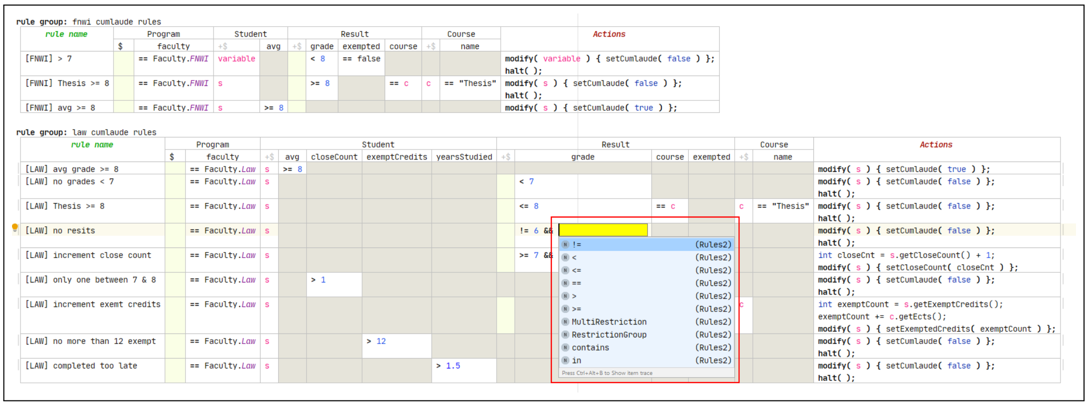


FIGURE 4.14: Spreadsheet projection

In this design each row is a rule, each column is for a variable or a property of a fact. If a property is selected then the selection criteria is in the appropriate cell. Unselected cells are indicated by a grey/beige color. The RHS of the rule appears in the Actions column. To add as yet unused facts or fact properties, or remove existing ones, can be achieved with intentions, as shown in figure 4.15.

This Design also allowed us to have more than one selector for the same FactProperty, this being important for our host organizations code. This is demonstrated in the figure 4.16.

This design was examined by experts in the questionnaire.

Here ends our experiments in the Drools-lite language.

4.2.3 Wireframe

After brainstorming several ideas to present as wireframes to experts as possible projectional aids to understanding we chose two. We discuss them briefly in this section.

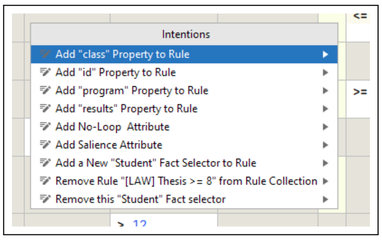


FIGURE 4.15: Intention

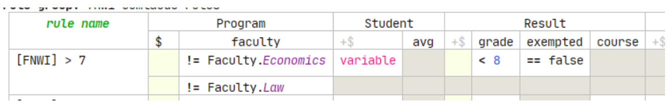


FIGURE 4.16: Two of same property

Truth Table

We decided to produce this example as we had had experience of building truth tables to confirm the validity of Drools rules in our work, thus we had an n of 1 whom we knew would find this projection useful.

The truth table seemed apt for the LHS of the Drools rule as in essence it is a boolean function. Wittgenstein popularized the truth table in the Tractatus Logico-Philosophicus[119]. It is so widely used in mathematics and computer science, that we do not feel the necessity to explain its use further. Because of the combinatorial explosive nature of truth tables, with 2^n possible combinations, thus we would limit display to a max of 6 variabiables and only show the paths that lead to the RHS being executed.

FIGURE 4.17: Truth table projection

Figure 4.17 shows how we designed this to look. The user experience would be that the rule is selected and the developer presses the up and down arrow keys to step through the different true (highlighted in green) and false (highlighted in red) fact selections that result in a true outcome.

We presented this design to our experts through the questionnaire to be validated.

Circuit Diagram

Our last projection design wanted to present a part of projectional editing that we had heretofore only mad minimal use of. That is the use of Graphics which can be manipulated to change the AST.

We chose a logic circuit. The logic circuit represents a boolean operations as NOT, OR, XOR and AND Gates with their inputs and outputs being inputs to other gates. In our design, shown in figure 4.18 the input wires to the gates are the Facts or FactProperties referenced in the LHS.

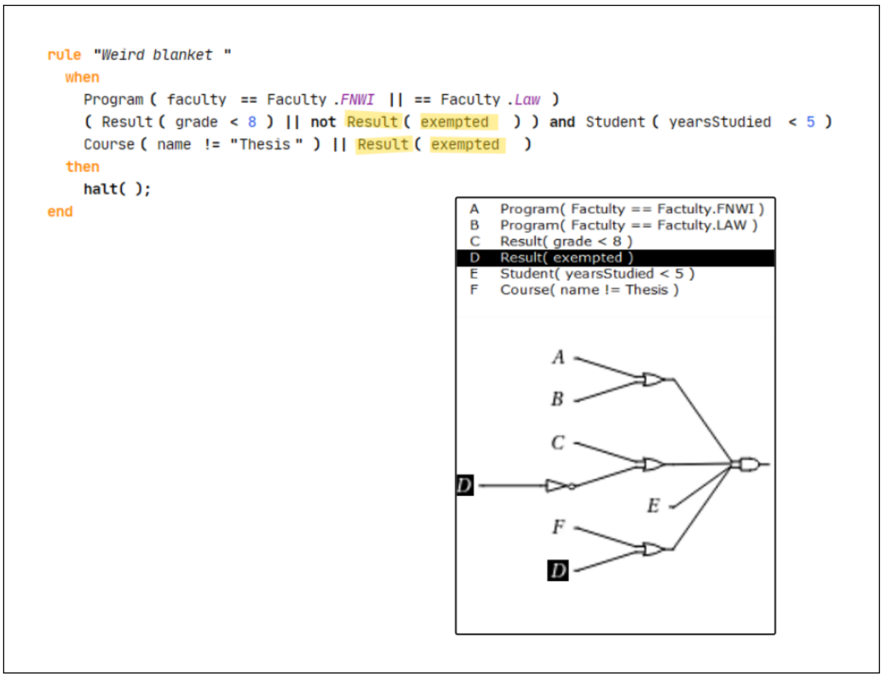


FIGURE 4.18: Circuit diagram projection

The user experience is that once the rule is selected, the developer, by pressing the up and down arrow keys, can step through the different fact selections (highlighted in yellow) and shown in the circuit diagram, thus showing how the facts relate to each other.

This design is also presented in the questionnaire for validation.

4.3 Results: Survey

4.3.1 Population Selection

We initially had two sources for our Experienced Drools users to send our survey to. From LinkedIn we selected users who were at one degree of separation from us and listed Drools in their skills. From StackOverflow we selected users who had asked or answered questions about Drools.

As in these two websites the users do not tend to list their contact details, some investigation was required. From the initial selection, whose size we did not record we harvested email accounts, and failing that twitter accounts.

A few days into our survey we read a paper that described the use of academic papers as a population of expertise. We used Google Scholar to look up Drools papers from the previous 2 years. After skimming the papers to ensure that it was specifically about or using the Drools language we harvested emails

On the second and fourth day of the survey two subjects forwarded the weblink to the survey to mailing lists. One, a developer from the core Drools team, sent it to a list of known Drools consultants. The other sent it internally in his company. both the subjects who sent the survey to their mailing list forwarded links to version C of the survey.

We had created 4 versions of the questionnaires to a combat single source bias. We distributed the surveys to the subjects harvested from LinkedIn and StackOverflow evenly. Because of the overrepresentation of Survey C, we distributed the subjects harvested from academic papers evenly over Surveys A, B and D.

The collection result can be seen in figure 4.19. What we see here is that the method of collection did not have much of an impact on return rates. whilst StackOverflow had a higher rate, the number of people contacted was so small that a small addition of respondents has an outsized effect on the proportion.

The first three pie charts represents the collection methods over which we had control. These three represented 24 of our 30 completed questionnaires. The last pie chart represents 6 completed and 4 partially completed questionnaires, that were returned from the surveys sent on by our initial participants. We do not know the size of the starting population of these lists. Thus this pie chart only shows the ratio of partial to completed results.

In summary, a survey reached known 154 participants, of which 24 completed it, for a Response Rate of 15.5%. In addition, an unknown amount of participants were reached through mailing lists, returning a further 6 completed surveys.

4.3.2 Participant Demography

Responses came from around the world. Figure 4.20 shows the location of the respondents were concentrated in Europe, the exceptions being the USA, Israel, and Singapore. Italy and the Netherlands provided the largest number of responses, with 7 and 5 respectively.

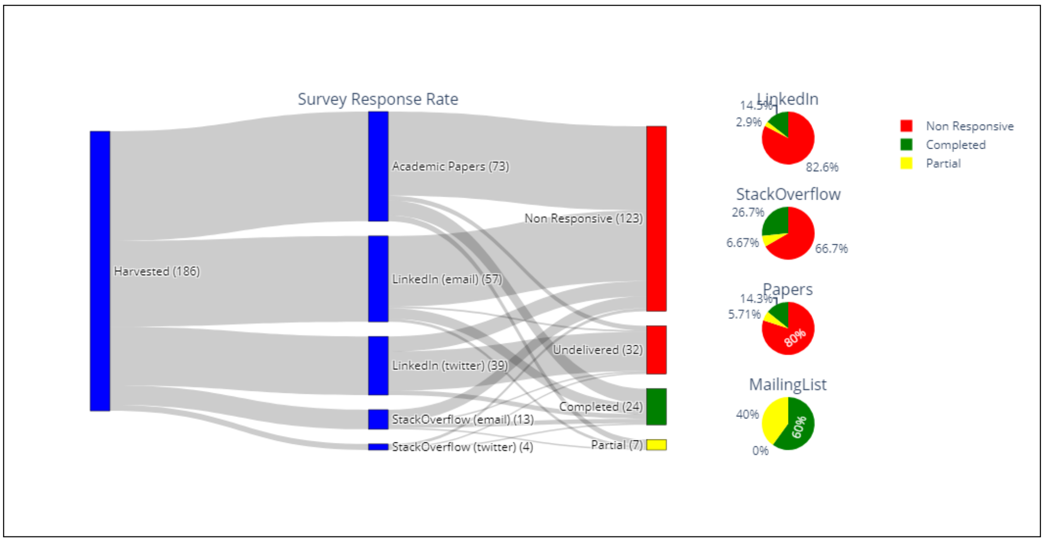


FIGURE 4.19: Survey participants

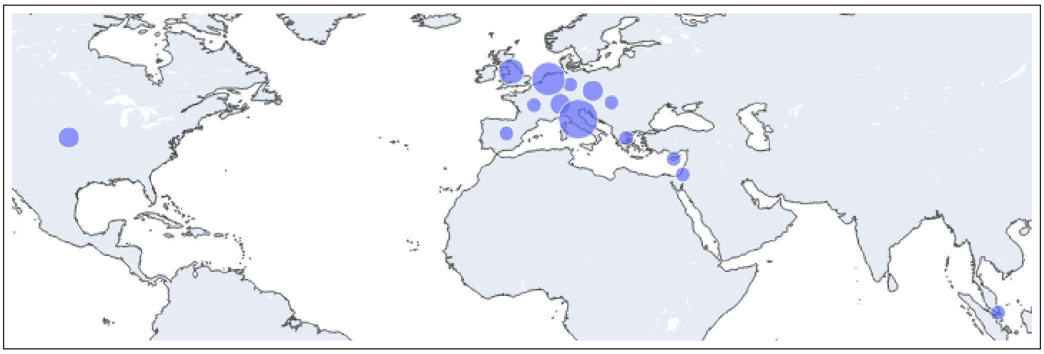
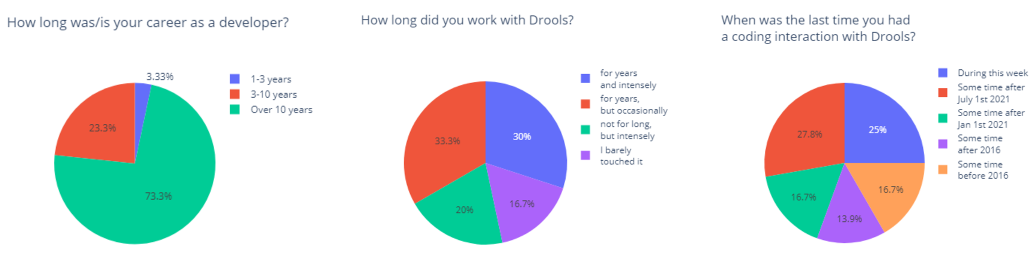


FIGURE 4.20: Survey locations



(A) SE experience (B) Drools experience (C) Recent use

FIGURE 4.21: Subject experience

The experience of our subjects was quite high. As can be seen in 4.21a, most of our subjects have over 10 years programming experience. 17% of our recipients had a low experience of Drools, and 30% were very experienced, as shown in figure 4.21b. Figure 4.21c reports that over half of our recipients have used Drools in the previous 6 weeks with only 17% not having used Drools for more than 5 years.

Half of our subjects reported only ever using one editor for Drools, with the slight majority of those only using Eclipse. Eclipse also had the most instances of reporting of having been used, out of the 55 instances of editors reported as being used, 20 of those were Eclipse. There was a, to us, surprising diversity of tools being used. The purpose of this section was to be able to calibrate responses against exposure to IDEs with greater Drools Support. The wide diversity of editor usage and high incidence of multiple editor usage means that these answers are not suitable for use in the sub-categorisation of responses. The distribution of usage is shown in figure 4.22.

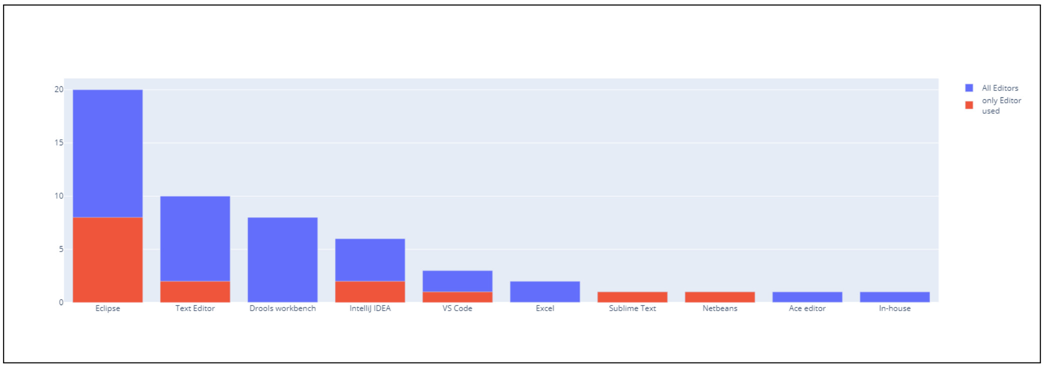


FIGURE 4.22: Editors used

4.3.3 Question Analysis

Grouping

When displaying subtypes we shall split into groups. The source of the response will create a pseudo cross-section of our participants. The 10 who were contacted through academic papers will be considered our academics. The remaining 20 will be considered practitioners.

The next grouping will be on Drools Experience. The 9 who replied they had used drools “for years and intensely” are categorised as experts. The 16 who either answered “for years, but occasionally” or “not for long, but intensely” are categorised as seniors. The 5 who answered “I barely touched it” are categorised as novices.

Another grouping will be on recency of use. The 12 who have used Drools in 2021 will be categorised as current users. The remaining 18 as past users.

To remove some bias in the questionnaires we changed question order, order of projections, and which rulesets were used. When a question is effected by this, then this will also be displayed.

Display

Until now, the choice of chart to display the survey outcome has been based on a feeling rather than research. For the remainder of this section we will rely on the advice of our predecessors.

The remainder of this Results sections, whilst displaying results that regard our Likert scaled questions, we will be following the advice of Robbins et.al.[120] by using diverging stacked bar charts, with counts added. This style allows the evaluation of subclasses results. The addition of counts makes it easier to spot when the results are skewed by small numbers.

In our charts we will take positive scores as being right of the center line and neutral and negative scores as being left. With our particular question design, we are unable to tell if the neutral responses are substantive or hidden non responses[121].

Grouping Analysis

Our data does not fall under a normal distribution, so we will be using a nonparametric test. Vaus[76] advises that when analysing Ordinal variables with nominal grouping variables then the statistical methods one should use for checking for differences between groupings would be the Mann-Whitney[122] or the Kruskal Wallis[123] tests. In the grouping section we had one group of only 5 participants, the novices. As this group is too small to think of using in analysis we will ignore it. This means that all of the groupings we have will be divided in two. As Mann-Whitney is a specialisation of Kruskal Wallis for two groups, this is the analysis we will use.

For all our analyizes our null hypothesis, H_0, is that there is no between the ranks of the two groups. Our alternative hypothesis, H_1, is there is a difference between the ranks of the two groups. Our alpha, or significance, level will be 0.05, for no other reason than it seems to be a mutually agreed upon value within the statistical community, and justifying a different significance would take more time than is warranted for the value these analysizes will give. Our sample size is greater than 20, thus we can use a Z distribution.

To work out out Z score, we have an alpha level of 0.05 and a 2 tailed test. We could work the distribution out by using the following formula:

$$U_1 = n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

However it is easier to look up in a Z table. Thus, we have an area in body of 0.9750, which correlates in the Z Table to a z value of 1.96. Thus our decision rule is, If our z is less than -1.96, or greater than 1.96, we reject our null hypothesis. [1]

[1] We carried out the calculations using the Mann-Whitney U Test calculator at https://www.socscistatistics.com/tests/mannwhitney/default2.aspx

Question 1, 2 and 3: First Impressions

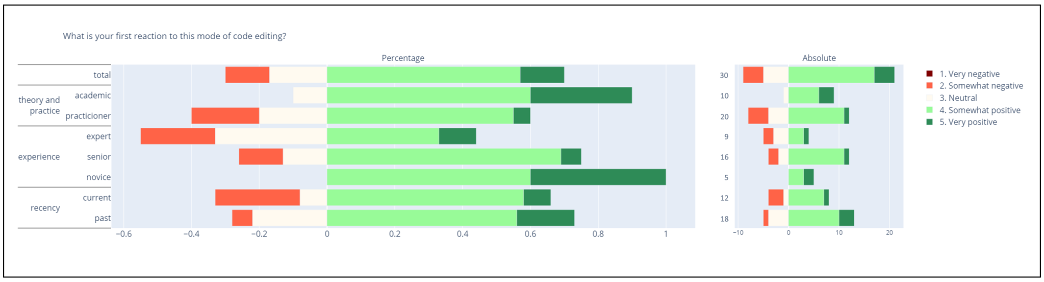


FIGURE 4.23: Question 1 - first impressions

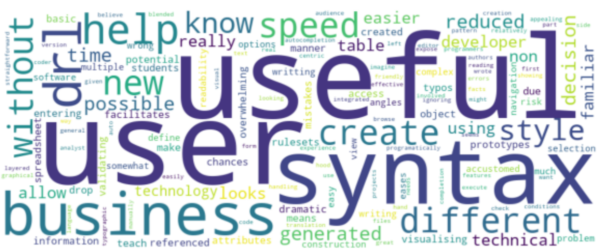
Group Comparison	Critical U	U-value	z-Score	p-value	Hypothesis
Academic vs Practitioner	55	54.5	-1.97974	0.0477	$\mathbf{H_1}$
Expert vs Senior	37	55	0.93413	0.35238	H_0
Current vs Past	61	91	0.6985	0.48392	H_0

TABLE 4.3: Mann-Whitney question 1 - first impressions

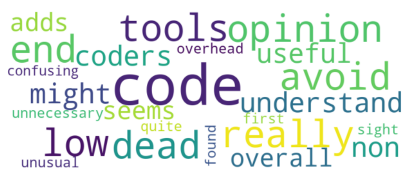
This question shows the subject an example of projectional editing a Drools file alongside a table projection, as an animated GIF, along with an explanation. Then she is asked her first reaction. The chart in figure 4.23 shows the outcomes. Eyeballing the chart, an observation that can be taken is that the novice and the academics (where there is a lot of crossover), found the initial presentation more positive than the experienced practitioners.

Table 4.3, shows that there is a significant difference between how academics and practitioners first react to seeing the projectional editing example. Here, the academics have a far more positive view of the example than the practitioners.

In general, we see an overwhelmingly positive response. There were 5 times as many positive (21) then negative (4) responses. Those who had a positive or negative responses were directed to answer the open questions "Q2. How would this coding style be useful to your interactions with Drools?" and "Q3. What do you find negative with this style of coding?" respectively. Figure 4.24 show a not very useful, but funky looking visualization of the subjects responses.



(A) Positive



(B) Negative

FIGURE 4.24: Initial thoughts

Amongst the positive comments it appears the the subjects had all picked up on many of the advantages that projectional editing brings. The ones that got the most mentions, using other words, were exploratory coding, correctness by construction, and multiple viewpoints. It was also noted, with the projections shown, that development could be quicker and easier to check.

Amongst the, very few, negative comments, they discussed the failures of no/low code solutions, that the view was confusing and they felt it added unnecessary overhead.

Question 4 and 5: Interpret Projection

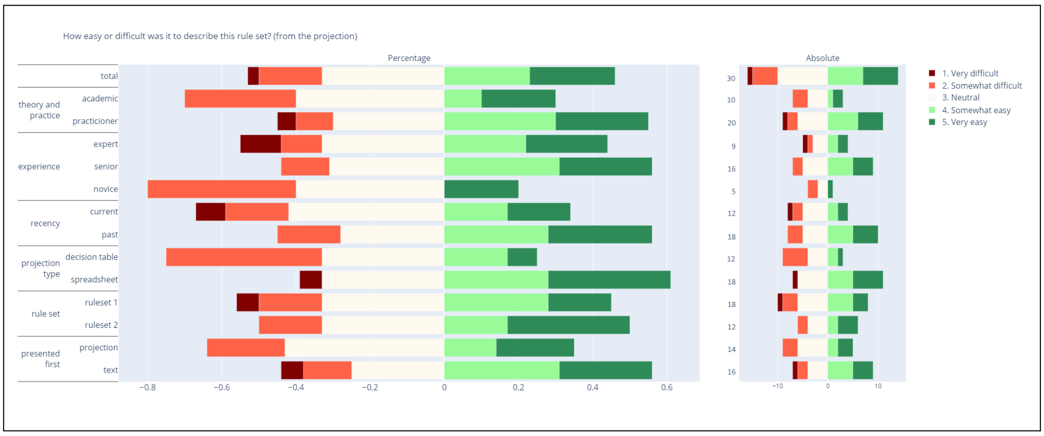


FIGURE 4.25: Question 5 - interpret projection

Group Comparison	Critical U	U-value	z-Score	p-value	Hypothesis
Academic vs Practitioner	55	77	0.98987	0.32218	H_0
Expert vs Senior	37	61.5	0.56614	0.56868	H_0
Current vs Past	61	82.5	1.05833	0.28914	H_0
Decision Table vs Spreadsheet	55	61	2.2225	0.02642	$\mathbf{H_1}$
Ruleset 1 vs Ruleset 2	61	92	0.65617	0.50926	H_0
Projection first vs Text first	64	97	0.60277	0.5485	H_0

TABLE 4.4: Mann-Whitney question 5 - interpret projection

This question asks the subject to describe the meaning of the projection and then describe how hard it was to do that. Very few people described the meaning of the projection well.

The chart in figure 4.25 shows the outcomes. Looking at the chart, it appears obvious that there is a difference in confidence of the subjects between the different projections presented. The participants believed they understood the spread sheet style projection better then the decision table projection.

This observation is confirmed in the Mann-Whitney analysis, shown in table 4.4. Otherwise, no other grouping differed significantly from the general population.

The ratio of those thinking it was easy to those thinking it was hard to understand the projection was 2:1, 14 to 6.

There was one extreme response, which was negative. This came from an expert practitioner who saw the text version before seeing the projection.

Question 6 and 7: Interpret Text

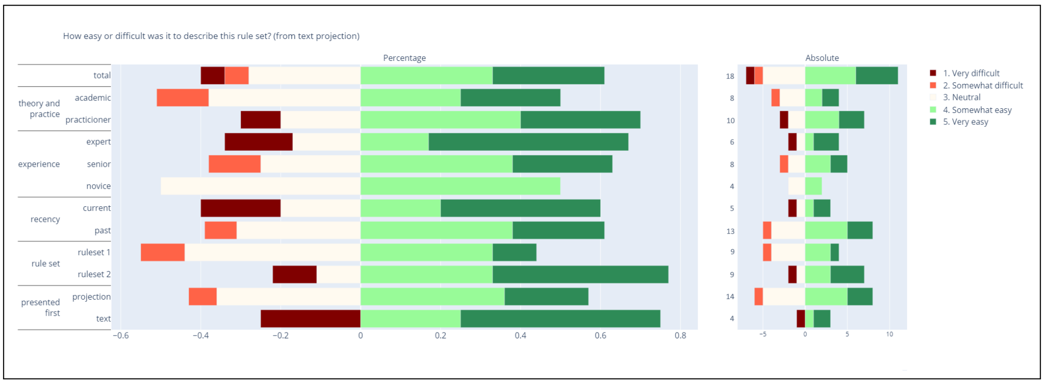


FIGURE 4.26: Question 7 - interpret text

Group Comparison	Critical U	U-value	z-Score	p-value	Hypothesis
Academic vs Practitioner	17	34	0.48869	0.62414	H_0
Expert vs Senior	8	20.5	-0.3873	0.69654	H_0
Current vs Past	12	31.5	-0.04929	0.96012	H_0
Ruleset 1 vs Ruleset 2	17	24.5	-1.36868	0.17068	H_0

TABLE 4.5: Mann-Whitney question 7 - interpret text

This question asks the subject to interpret a rule set that is presented in a Drools style text projection. The purpose of this question was three fold. First, to calibrate how well the subject really understood drools. Second, for a comparison with with a later projection. Finally, to calibrate whether and how much easier the text was than the projection to those used to seeing the text version.

Unfortunately, we assume due to the questionnaire design, there was a large number of non-respondents to this question. 12 of the 30 respondents did not answer this question. All 12 of these were from the 16 that were presented the text projection before the tabular projections.

Reporting on 18 responses has a lot less validity. With that in mind in figure 4.26, we can see a much higher confidence in the easiness of understanding the rule set. We see a 5:1 ratio of greater belief in the participants that it was easy to understood the meaning of the text projections. This proportion is significantly higher than those who thought it was easy to understood the tabular projections.

The Mann-Whitney table does not show any significant differences in any of the groups. We are unable to report on the difference between those who were presented the text projection first or the tabular projection first. The 4 participants responding from the group who were presented text first, was too low to give a meaningful score.

Question 8: Compare Projections

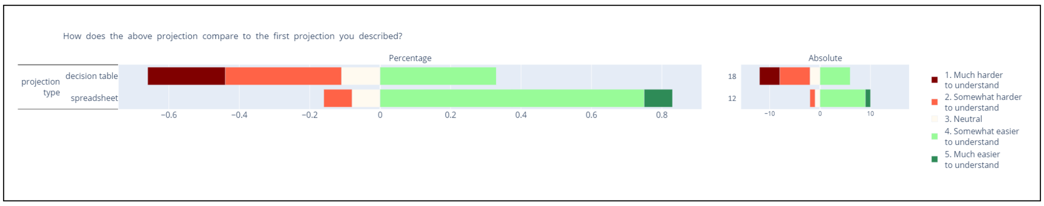


FIGURE 4.27: Question 8 - compare projections

Group Comparison	Critical U	U-value	z-Score	p-value	Hypothesis
Decision Table vs Spreadsheet	61	45	-2.64584	0.00804	$\mathbf{H_1}$

TABLE 4.6: Mann-Whitney question 8 - compare projections

This question asked the participant to compare the two tabular projections. This question was to calibrate whether one projection was considerably worse than the other and whether that would affect the comparison with text.

Looking the chart in figure 4.27, we have ignored all bars except the difference between the projections presented. The way the question is constructed the other groupings do not help.

It is obvious in this chart and confirmed in the Mann-Whitney analysis shown in table 4.6, that the spreadsheet presentation is considered far easier to understand than the decision table. This result correlates well with the difference in understanding found previously in question 5.

Question 9: Compare Projection to Text

FIGURE 4.28: Question 9 - compare projections with text

Group Comparison	Critical U	U-value	z-Score	p-value	Hypothesis
Academic vs Practitioner	52	85.5	-0.41295	0.6818	H_0
Expert vs Senior	34	67.5	0.02981	0.97606	H_0
Current vs Past	55	88	0.47194	0.63836	H_0
Decision Table vs Spreadsheet	55	89.5	-0.40452	0.68916	H_0
Ruleset 1 vs Ruleset 2	55	94.5	-0.17979	0.85716	H_0

TABLE 4.7: Mann-Whitney Question 9 - compare projections with text

This question asked our subjects to compare a tabular projection with the text projection. There was one participant that chose not to answer this question.

This result, shown in figure 4.28, was pretty definitive. The subjects found very much that the textual projections were more understandable than the tabular projections. The ratio of harder to easier to understand was 3:2, with 12 subjects finding the Text projection easier to understand and 8 finding the projection easier. Table 4.7 shows this was independent of any other factors.

Question 10: Truth Table Validation

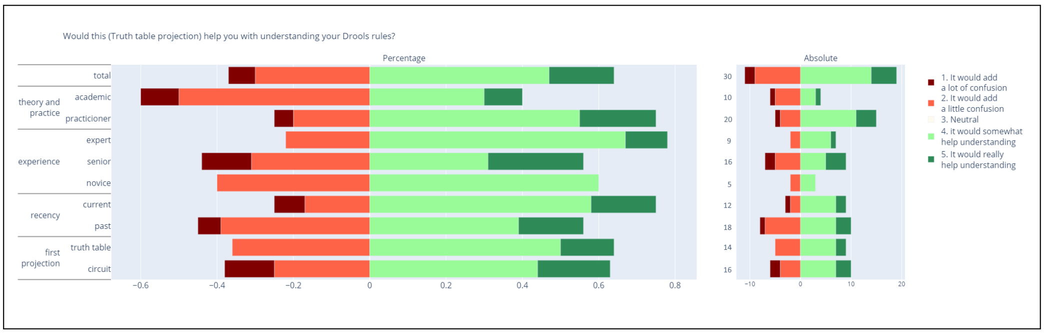


FIGURE 4.29: Question 10 - truth table

Group Comparison	Critical U	U-value	z-Score	p-value	Hypothesis
Academic vs Practitioner	55	65	1.5178	0.12852	H_0
Expert vs Senior	37	64	-0.4246	0.67448	H_0
Current vs Past	61	93	-0.61383	0.54186	H_0
Truth table first vs Circuit first	64	108.5	-0.12471	0.90448	H_0

TABLE 4.8: Mann-Whitney Question 10 - truth table

This question presented the subject with a wireframe of a truth table and asked if it would help them with understanding.

Every subject had a positive or negative view of the projection. 0 of the 30 subjects gave a neutral result. We found this a very unlikely result.

There was a net positive view of this projection, with a little less than 2:1 finding it more helpful (19) than confusing (11).

On first view of the chart in figure 4.29, it seems that academics have a much more negative view of this projection, and experts a more positive outlook. However, when the figures are examined under Mann-Whitney, as seen in table 4.8, these differences were not statistically significant.

Question 11: Circuit Diagram Validation

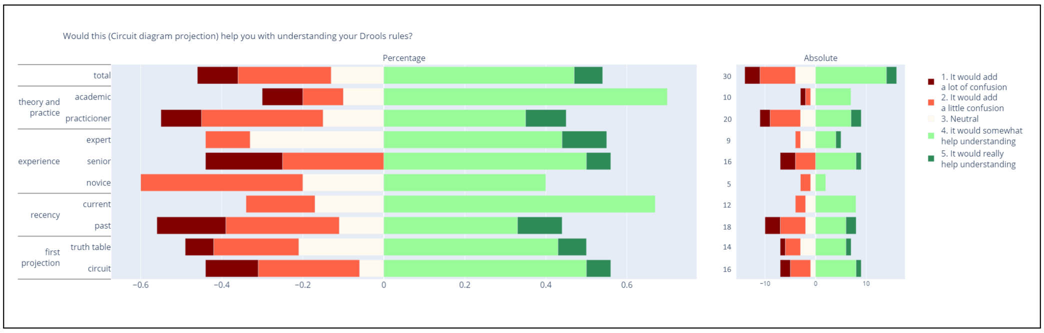


FIGURE 4.30: Question 11 - circuit diagram

Group Comparison	Critical U	U-value	z-Score	p-value	Hypothesis
Academic vs Practitioner	55	83	-0.7259	0.4654	H_0
Expert vs Senior	37	58.5	-0.73598	0.4593	H_0
Current vs Past	61	83	-1.03717	0.29834	H_0
Truth table first vs Circuit first	64	110	-0.06236	0.95216	H_0

TABLE 4.9: Mann-Whitney Question 11 - circuit diagram

This question presented the subject with a wireframe of a circuit diagram and asked if it would help them with understanding.

The view of the circuit diagram was a a net positive, with a ratio of 3:2 (16 positive, 10 negative).

Question 15: Closing Remarks

Our last question asks for any last comments. 16 participants decided to add some words.

We ran sentiment analysis on these 16 comments, using the same service and similar code as we used in our SLR research, as described in section 3.1.6. 9 were considered positive, 6 mixed and 1 negative.

Figure 4.31 shows a another pretty, but not too analytical word cloud.

The word that pops out, DMN, here is indicative of a few comments. DMN is a tool for working with drools as a non developer. There were some comments that our research should look in this direction.

The feeling was that the projections would give more advantage to non-developers.

There were some hints as to how to improve our projections or other projections to try. Others suggested that the projections cannot capture some of the complexity of the rules that exist.

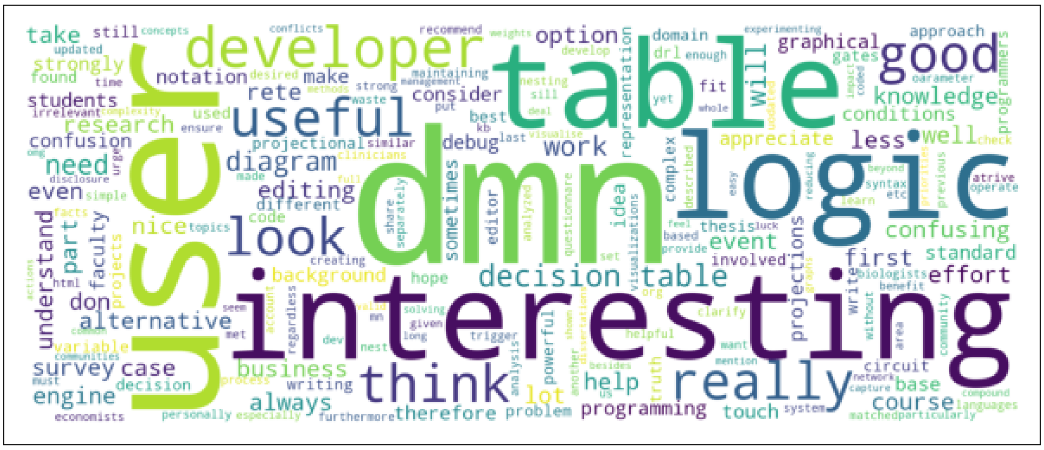


FIGURE 4.31: Question 15 - responses

4.3.4 Summary Analysis

The goal of this survey was to find if users of drools would find the projections we used useful. The outcome seems to be yes, they seem useful, but not better than the text that they are currently used to.

We tried to control for different groupings that my effect the responses, and in most cases we found no significant differences in the groups we presented.

There was a significant difference between the projections presented, with the sample in general preferring the spreadsheet projection over the decision table. However, this was not what we were measuring for and even that is debatable as in question 9. When comparing text to projection, the outcome for both tabular projections favoured text, but with a slightly lower net negative for the decision table comparison, 7:5 compared to 3:2.

Chapter 5

Discussion

5.1 Discussion - Systematic Literature Review

We now examine threats to the construct, the internal and external validity of our systematic literature review, its reliability, and areas of improvement.

- survey selection
- quality assessment
- data extraction
- search phrase choice
- that sentiment analysis is invalid for scientific papers
- azure sentiment analysis is not calibrated for scientific papers

5.1.1 Threats to Validity

As discussed in their tertiary study of SLRs da Silva et al.[124], one of the main problems of SLRs in Software engineering is a focus on practice and not experimentation. Because of the nature of the subject area we will be making this same shortfall. We feel that we have fully addressed their other concerns of SLRs not assessing the quality of our primary studies, bad integration and lack of guidelines.

As with all SLRs the main threats to validity are incomplete set of studies, due to an insufficient search strategy, researcher bias in paper selection and inaccuracy in data extraction. In our study quality assessment we made use of Runeson et al.'s[125] four suggested limitations of studies, namely construct validity, internal validity, external validity, and reliability. It is only fair that we point this towards our own study.

Construct Validity

Regarding construct validity, i.e. whether our research questions match the research subjects methods and measures. Whilst no measurement system is perfect, some are much further from perfect than others.

For the construct to be valid we need to present the best available evidence. The nature and modernity of the Projectional Editing might mean that there is plenty of good evidence available in Grey literature and Industrial Articles. This under representation of actual, but non-academic studies could lead to a false positive or negative for some of the questions, leading to errors in recommendations.

There maybe things that influence the best evidence such as who is funding the study. Is a researcher working or consulting at a Projectional Editing product supplier, and will this skew results.

Are some projectional editors being ignored because of the preference for English papers only? The focus on English language papers might be biased against projectional editors aimed at non-English speaking markets.

Internal Validity

Internal validity, or the causal relationships, does one factor cause an effect or are both factors influenced by something unseen. On the surface this seems like it would not affect and SLR.

Selection bias - "projectional editing"

External Validity

External validity is the ability to generalize the findings.

Reliability

Reliability is how the data and the analysis is dependant on specific researchers. Here we are presented with a very credible threat in that this research was carried out by a single researcher.

Whilst measures were put in place to try and mitigate this, the reliance on a single person's judgement of the underlying studies, leaves the door to bias wide open. Another threat is the use of thematic review and narrative review. Both of these can be subjective and therefore difficult to reproduce.

In a replication the papers may be met in a different order leading to different predominant themes. A big thread to reliability was the [TO DO -]. This meant that the selection by title and subsequent selection by abstract were so long winded that these took place over multiple days. Thus there is an impact of bias in time of day and time in the session that might effect whether a title that could be considered marginal would be in or out.

Repeatability vs Reproducibility

Greenhalgh et al.[126] in their review of where papers come from in systematic reviews found that 24% of papers come from "personal knowledge or personal contact". for the sake of reproducibility we decided not to hunt down relevant papers from knowledgeable people in the field, as this would require anyone trying to reproduce this study to ask the same people at the same knowledge level as us. This would be impossible as the only person we know who could help us with this is our academic advisor, and he (in our future and your present) will have read this paper, changing his knowledge base forever. by making this chase we risk missing out a quarter of our completeness.

Method Improvement

5.2 Discussion - Survey

5.2.1 Threats to Validity

Construct Validity

Internal Validity

External Validity

Reliability

Repeatability vs Reproducibility

Method improvement

Chapter 6

Implications to research and practice

6.1 Implications to Research

6.2 Future Research Directions

6.3 Implications to Practice

Chapter 7

Conclusion

We have built our projections as an aid to the understanding of Drools rules. This DSL extension includes many different ways to look at and interact with large code bases, as well as presenting options to deal with the complexity of individual rules. This means that they must be [TODO: PROPERTIES THAT AIDS UNDERSTANDING]. Our questionnaires show that we have reached that aim. Since developing our projections we have used them to model complex rules in our host organization.

Two factors lead to our success. First was the flexibility and extensibility of the MPS tool which presented the ability to develop and extend DSLs very efficiently. If we had tried this project without this tooling we would have [TODO: Finish our thoughts] Second [TODO: Finish our thoughts]

In this paper we described our work with first translating the Drools DSL into a projectional language followed by our explorations of projections. We discussed the advantages and disadvantages of the different projections we created and analysed experienced developers reactions to them.

Whilst we are convinced our projections [TODO: finish our thoughts]

Bibliography

[1] C. 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.

[2] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information.," *Psychological review*, vol. 63, no. 2, p. 81, 1956.

[3] C. J. Date, *What not how: the business rules approach to application development*. Addison-Wesley Professional, 2000.

[4] M. Fowler, *Should i use a rules engine?* https://martinfowler.com/bliki/RulesEngine.html, Accessed: 2021-07-18, 2009.

[5] P. Jackson, "Introduction to expert systems," 1986.

[6] E. H. Shortliffe, "Mycin: A rule-based computer program for advising physicians regarding antimicrobial therapy selection.," STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE, Tech. Rep., 1974.

[7] *CLIPS product page*, http://www.clipsrules.net/, Accessed: 2021-07-17.

[8] *Drools product page*, https://www.drools.org/, Accessed: 2021-07-17.

[9] *BizTalk product page*, https://docs.microsoft.com/en-gb/biztalk/, Accessed: 2021-07-17.

[10] *IBM WebSphere JRules product page*, https://www.ibm.com/docs/en/iis/11.7?topic=applications-websphere-ilog-jrules, Accessed: 2021-07-17.

[11] *OpenRules product page*, https://openrules.com/, Accessed: 2021-07-17.

[12] D. Sottara, P. Mello, and M. Proctor, "A configurable rete-oo engine for reasoning with different types of imperfect information," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 11, pp. 1535–1548, 2010.

[13] V. Donzeau-Gouge, G. Huet, G. Kahn, and B. Lang, "Programming environments based on structured editors: The mentor experience," Tech. Rep., 1980.

[14] R. Medina-Mora and P. H. Feiler, "An incremental programming environment," *IEEE Transactions on Software Engineering*, no. 5, pp. 472–482, 1981.

[15] D. Notkin, "The gandalf project," *The Journal of systems and software*, vol. 5, no. 2, pp. 91–105, 1985.

[16] T. Teitelbaum and T. Reps, "The cornell program synthesizer: A syntax-directed programming environment," *Communications of the ACM*, vol. 24, no. 9, pp. 563–573, 1981.

[17] T. W. Reps and T. Teitelbaum, *The Synthesizer Generator: A system for constructing language-based editors*. Springer Science & Business Media, 2012.

[18] C. Simonyi, "The death of computer languages, the birth of intentional programming," in *NATO Science Committee Conference*, Citeseer, 1995, pp. 17–18.

[19] S. Dmitriev, "Language oriented programming: The next programming paradigm," *JetBrains onboard*, vol. 1, no. 2, pp. 1–13, 2004.

[20] S. Erdweg, T. Van Der Storm, M. Völter, L. Tratt, R. Bosman, W. R. Cook, A. Gerritsen, A. Hulshout, S. Kelly, A. Loh, *et al.*, "Evaluating and comparing language workbenches: Existing results and benchmarks for the future," *Computer Languages, Systems & Structures*, vol. 44, pp. 24–47, 2015.

[21] *Más post mortem*, https://medium.com/@dslmeinte/post-mortem-for-m%C3%A1s-aeca7542c4c8, Accessed: 2021-09-07.

[22] *Whole Platform*, https://whole.sourceforge.io/, Accessed: 2021-09-07.

[23] D. H. Lorenz and B. Rosenan, "Cedalion: A language for language oriented programming," in *Proceedings of the 2011 ACM international conference on Object oriented programming systems languages and applications*, 2011, pp. 733–752.

[24] B. Hempel, J. Lubin, G. Lu, and R. Chugh, "Deuce: A lightweight user interface for structured editing," in *Proceedings of the 40th International Conference on Software Engineering*, 2018, pp. 654–664.

[26] *Google blockly product page*, https://developers.google.com/blockly/, Accessed: 2021-09-08.

[27] J. Klimeš, "Domain-specific language for learning programming," 2016.

[28] D. Ratiu, V. Pech, and K. 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.

[29] M. Völter and E. Visser, "Language extension and composition with language workbenches," in *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, 2010, pp. 301–304.

[30] M. Voelter, J. Siegmund, T. Berger, and B. Kolb, "Towards user-friendly projectional editors," in *International Conference on Software Language Engineering*, Springer, 2014, pp. 41–61.

[31] M. Hosseinkord, G. Dulai, N. Osmani, and C. K. Anand, "Code and structure editing for teaching: A case study in using bibliometrics to guide computer science research," *arXiv preprint arXiv:2107.09038*, 2021.

[32] M. Voelter, S. Benz, C. Dietrich, B. Engelmann, M. Helander, L. C. Kats, E. Visser, G. Wachsmuth, *et al.*, *DSL engineering: Designing, implementing and using domain-specific languages*. dslbook. org, 2013.

[33] D. Ratiu, M. Gario, and H. Schoenhaar, "Fasten: An open extensible framework to experiment with formal specification approaches," in *2019 IEEE/ACM 7th International Conference on Formal Methods in Software Engineering (FormaliSE)*, IEEE, 2019, pp. 41–50.

[34] D. Ratiu, B. Schaetz, M. Voelter, and B. Kolb, "Language engineering as an enabler for incrementally defined formal analyses," in *2012 First International Workshop on Formal Methods in Software Engineering: Rigorous and Agile Approaches (FormSERA)*, IEEE, 2012, pp. 9–15.

[35] T. Berger, M. Völter, H. P. Jensen, T. Dangprasert, and J. Siegmund, "Efficiency of projectional editing: A controlled experiment," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2016, pp. 763–774.

[36] P. Vysok, P. Parizek, and V. Pech, "Ingrid: Creating languages in mps from antlr grammars," 2018.

[37] V. Pech, "Jetbrains mps: Why modern language workbenches matter," in *Domain-Specific Languages in Practice*, Springer, 2021, pp. 1–22.

[38] M. Voelter and S. Lisson, "Supporting diverse notations in mps'projectional editor.," in *GEMOC MoDELS*, 2014, pp. 7–16.

[39] M. Voelter and K. Solomatov, "Language modularization and composition with projectional language workbenches illustrated with mps," *Software Language Engineering, SLE*, vol. 16, no. 3, 2010.

[40] M. Voelter, A. v. Deursen, B. Kolb, and S. Eberle, "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.

[41] M. Voelter, "Embedded software development with projectional language workbenches," in *International Conference on Model Driven Engineering Languages and Systems*, Springer, 2010, pp. 32–46.

[42] S. M. Guttormsen, A. Prinz, and T. Gjøsæter, "Consistent projectional text editors.," in *MODELSWARD*, 2017, pp. 515–522.

[43] M. Voelter and B. Merkle, "Domain specific: A binary decision?" In *Proceedings of the 10th Workshop on Domain-Specific Modeling*, 2010, pp. 1–6.

[44] A. Wortmann and M. Beet, "Domain specific languages for efficient satellite control software development," *OHB System AG*, 2016.

[45] M. Voelter, D. Ratiu, B. Kolb, and B. Schaetz, "Mbeddr: Instantiating a language workbench in the embedded software domain," *Automated Software Engineering*, vol. 20, no. 3, pp. 339–390, 2013.

[46] C. Simonyi, M. Christerson, and S. Clifford, "Intentional software," in *Proceedings of the 21st annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, 2006, pp. 451–464.

[47] M. Voelter, T. Szabó, S. Lisson, B. Kolb, S. Erdweg, and T. Berger, "Efficient development of consistent projectional editors using grammar cells," in *Proceedings of the 2016 ACM SIGPLAN International Conference on Software Language Engineering*, 2016, pp. 28–40.

[48] V. Pech, A. Shatalin, and M. 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.

[49] M. Voelter, J. Warmer, and B. Kolb, "Projecting a modular future," *IEEE Software*, vol. 32, no. 5, pp. 46–52, 2014.

[50] D. Ratiu, H. Nehls, and J. Michel, "Taming the software development complexity with domain specific languages," *Modellierung 2018*, 2018.

[51] M. Voelter, Z. Molotnikov, and B. Kolb, "Towards improving software security using language engineering and mbeddr c," in *Proceedings of the Workshop on Domain-Specific Modeling*, 2015, pp. 55–62.

[52] M. Voelter, B. Kolb, K. Birken, F. Tomassetti, P. Alff, L. Wiart, A. Wortmann, and A. Nordmann, "Using language workbenches and domain-specific languages for safety-critical software development," *Software & Systems Modeling*, vol. 18, no. 4, pp. 2507–2530, 2019.

[53] S. Meacham, V. Pech, and D. Nauck, "Adaptivevle: An integrated framework for personalized online education using mps jetbrains domain-specific modeling environment," *IEEE Access*, vol. 8, pp. 184 621–184 632, 2020.

[54] D. Pavletic, S. A. Raza, M. Voelter, B. Kolb, and T. Kehrer, "Extensible debuggers for extensible languages," *GI/ACM WS on Software Reengineering*, 2013.

[55] M. Voelter, "Language and ide modularization and composition with mps," in *International Summer School on Generative and Transformational Techniques in Software Engineering*, Springer, 2011, pp. 383–430.

[56] D. Ratiu, M. Voelter, Z. Molotnikov, and B. Schaetz, "Implementing modular domain specific languages and analyses," in *Proceedings of the Workshop on Model-Driven Engineering, Verification and Validation*, 2012, pp. 35–40.

[57] M. Voelter, D. Ratiu, B. Schaetz, and B. Kolb, "Mbeddr: An extensible c-based programming language and ide for embedded systems," in *Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity*, 2012, pp. 121–140.

[58] M. Voelter and E. Visser, "Product line engineering using domain-specific languages," in *2011 15th International Software Product Line Conference*, IEEE, 2011, pp. 70–79.

[59] M. Voelter, D. Ratiu, and F. Tomassetti, "Requirements as first-class citizens: Integrating requirements directly with implementation artifacts," in *Proceedings of ACES-MB Workshop*, Citeseer, 2013.

[60] E. Schindler, K. Schindler, F. Tomassetti, and A. Sutii, "Language workbench challenge 2016: The jetbrains meta programming system," in *LWC SLE 2016 Language Workbench Challenge, Splash2016, 39 October-4 November 2016, Amsterdam, The Netherlands*, 2016.

[61] A. Prinz, "Teaching language engineering using mps," in *Domain-Specific Languages in Practice*, Springer, 2021, pp. 315–336.

[62] F. Tomassetti and V. Zaytsev, "Reflections on the lack of adoption of domain specific languages.," in *STAF Workshops*, 2020, pp. 85–94.

[63] M. Fowler, *Language workbenches: The killer-app for domain specific languages?* http://martinfowler.com/articles/languageWorkbench.html, Accessed: 2021-02-02, 2005.

[64] B. A. Kitchenham, D. Budgen, and P. Brereton, *Evidence-Based Software Engineering and Systematic Reviews*. Chapman & Hall/CRC, 2015, ISBN: 1482228653.

[65] H. Zhang, M. A. Babar, and P. Tell, "Identifying relevant studies in software engineering," *Inf. Softw. Technol.*, vol. 53, no. 6, pp. 625–637, 2011.

[66] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, Association for Computing Machinery, 2014.

[67] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[68] A. O'Mara-Eves, J. Thomas, J. McNaught, M. Miwa, and S. Ananiadou, "Using text mining for study identification in systematic reviews: A systematic review of current approaches," *Systematic Reviews*, vol. 4, 2015.

[69] E. Akl, D. Altman, P. Aluko, L. Askie, D. Beaton, J. Berlin, B. Bhaumik, C. Bingham, M. Boers, A. Booth, I. Boutron, S. Brennan, M. Briel, S. Briscoe, J. Busse, D. Caldwell, M. Cargo, A. Carrasco-Labra, A. Chaimani, and C. Young, *Cochrane Handbook for Systematic Reviews of Interventions*. Oct. 2019.

[70] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele Univ.*, vol. 33, Aug. 2004.

[71] S. Keele *et al.*, "Guidelines for performing systematic literature reviews in software engineering," Technical report, Ver. 2.3 EBSE Technical Report. EBSE, Tech. Rep., 2007.

[72] S. Gregor, "The nature of theory in information systems," *MIS quarterly*, pp. 611–642, 2006.

[73] B. A. Kitchenham, D. Budgen, and P. Brereton, *Evidence-based software engineering and systematic reviews*. CRC press, 2015, vol. 4.

[74] *Axure product page*, `https://www.axure.com/`, Accessed: 2021-08-23.

[75] A. Bryman, *Social research methods*. Oxford university press, 2016.

[76] D. De Vaus and D. de Vaus, *Surveys in social research*. Routledge, 2013.

[77] R. Likert, "A technique for the measurement of attitudes.," *Archives of psychology*, 1932.

[78] J. S. Armstrong and T. S. Overton, "Estimating nonresponse bias in mail surveys," *Journal of marketing research*, vol. 14, no. 3, pp. 396–402, 1977.

[79] P. M. Podsakoff, S. B. MacKenzie, J.-Y. Lee, and N. P. Podsakoff, "Common method biases in behavioral research: A critical review of the literature and recommended remedies.," *Journal of applied psychology*, vol. 88, no. 5, p. 879, 2003.

[80] M. K. Lindell and D. J. Whitney, "Accounting for common method variance in cross-sectional research designs.," *Journal of applied psychology*, vol. 86, no. 1, p. 114, 2001.

[81] G. Gorrell, N. Ford, A. Madden, P. Holdridge, and B. Eaglestone, "Countering method bias in questionnaire-based user studies," *Journal of Documentation*, 2011.

[82] N. J. Goldstein, S. J. Martin, and R. Cialdini, *Yes: 50 scientifically proven ways to be persuasive*. Simon and Schuster, 2008.

[83] I. K. Crombie and B. J. Harvey, "The pocket guide to critical appraisal: A handbook for health care professionals," *Canadian Medical Association. Journal*, vol. 157, no. 4, p. 448, 1997.

[112] M. F. Cowlishaw, "Lexx—a programmable structured editor," *IBM Journal of Research and Development*, vol. 31, no. 1, pp. 73–80, 1987.

[113] A. Sarkar, "The impact of syntax colouring on program comprehension.," in *PPIG*, 2015, p. 8.

[114] D. Hou and D. M. Pletcher, "Towards a better code completion system by api grouping, filtering, and popularity-based ranking," in *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*, 2010, pp. 26–30.

[115] Y. Yoon, B. A. Myers, and S. Koo, "Visualization of fine-grained code change history," in *2013 IEEE Symposium on Visual Languages and Human Centric Computing*, IEEE, 2013, pp. 119–126.

[116] E. W. Dijkstra, "The humble programmer," *Communications of the ACM*, vol. 15, no. 10, pp. 859–866, 1972.

[117] U. W. Pooch, "Translation of decision tables," *ACM Computing Surveys (CSUR)*, vol. 6, no. 2, pp. 125–151, 1974.

[118] L. Bradley and K. McDaid, "Using bayesian statistical methods to determine the level of error in large spreadsheets.," in *2009 31st International Conference on Software Engineering-Companion Volume*, IEEE, 2009, pp. 351–354.

[119] L. Wittgenstein, *Tractatus logico-philosophicus.* Routledge, 2013.

[120] N. B. Robbins, R. M. Heiberger, *et al.*, "Plotting likert and other rating scales," in *Proceedings of the 2011 joint statistical meeting*, vol. 1, 2011.

[121] J. Blasius and V. Thiessen, "The use of neutral responses in survey questions: An application of multiple correspondence analysis," *JOURNAL OF OFFICIAL STATISTICS-STOCKHOLM-*, vol. 17, no. 3, pp. 351–368, 2001.

[122] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.

[123] W. H. Kruskal and W. A. Wallis, "Use of ranks in one-criterion variance analysis," *Journal of the American statistical Association*, vol. 47, no. 260, pp. 583–621, 1952.

[124] F. Q. Da Silva, A. L. Santos, S. Soares, A. C. C. França, C. V. Monteiro, and F. F. Maciel, "Six years of systematic literature reviews in software engineering: An updated tertiary study," *Information and Software Technology*, vol. 53, 2011.

[125] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, no. 2, 2009.

[126] T. Greenhalgh and R. Peacock, "Effectiveness and efficiency of search methods in systematic reviews of complex evidence: Audit of primary sources," *BMJ*, vol. 331, no. 7524, pp. 1064–1065, 2005.

Systematic Review Bibliography

[25] L.-E. Lafontant and E. Syriani, "Gentleman: A light-weight web-based projectional editor generator," in *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, 2020, pp. 1–5.

[84] M. Voelter, S. Koščejev, M. Riedel, A. Deitsch, and A. Hinkelmann, "A domain-specific language for payroll calculations: A case study at DATEV," 2021.

[85] J. Schröpfer, T. Buchmann, and B. Westfechtel, "A framework for projectional multi-variant model editors.," in *MODELSWARD*, 2021, pp. 294–305.

[86] J. Schröpfer, B. Westfechtel, and T. Buchmann, "A generic projectional editor for EMF models.," in *MODELSWARD*, 2020, pp. 381–392.

[87] A. Bucchiarone, K. Soysal, and C. Guidi, "A model-driven approach towards automatic migration to microservices," in *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, Springer, 2019, pp. 15–36.

[88] S. Meacham, V. Pech, and D. Nauck, "AdaptiveVLE: An integrated framework for personalized online education using MPS JetBrains domain-specific modeling environment," *IEEE Access*, vol. 8, pp. 184 621–184 632, 2020.

[89] L. Andersen, M. Ballantyne, and M. Felleisen, "Adding interactive visual syntax to textual code," *Proceedings of the ACM on Programming Languages*, vol. 4, no. OOPSLA, pp. 1–28, 2020.

[90] L. Addazi and F. Ciccozzi, "Blended graphical and textual modelling for uml profiles: A proof-of-concept implementation and experiment," *Journal of Systems and Software*, vol. 175, p. 110 912, 2021.

[91] S. Meacham, V. Pech, and D. Nauck, "Classification algorithms framework (CAF) to enable intelligent systems using JetBrains MPS domain-specific languages environment," *IEEE Access*, vol. 8, pp. 14 832–14 840, 2020.

[92] A. L. Furtado, "DSL based approach for building model-driven questionnaires," in *Enterprise Information Systems: 22nd International Conference, ICEIS 2020, Virtual Event, May 5–7, 2020, Revised Selected Papers*, Springer Nature, 2021, p. 458.

[93] T. Beckmann, "Efficient editing in a tree-oriented projectional editor," in *Conference Companion of the 4th International Conference on Art, Science, and Engineering of Programming*, 2020, pp. 215–216.

[94] D. Kolovos, A. De La Vega, and J. Cooper, "Efficient generation of graphical model views via lazy model-to-text transformation," in *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, 2020, pp. 12–23.

[95] A. Bucchiarone, A. Cicchetti, and A. Marconi, "Engineering gameful applications with MPS," in *Domain-Specific Languages in Practice*, Springer, 2021, pp. 227–258.

[96] D. Ratiu, A. Nordmann, P. Munk, C. Carlan, and M. Voelter, “FASTEN: An extensible platform to experiment with rigorous modeling of safety-critical systems,” in *Domain-Specific Languages in Practice*, Springer, 2021, pp. 131–164.

[97] J. Schröpfer and T. Buchmann, “Integrating UML and ALF: An approach to overcome the code generation dilemma in model-driven software engineering,” in *International Conference on Model-Driven Engineering and Software Development*, Springer, 2019, pp. 1–26.

[98] A. L. Santos, “Javardise: A structured code editor for programming pedagogy in java,” in *Conference Companion of the 4th International Conference on Art, Science, and Engineering of Programming*, 2020, pp. 120–125.

[99] E. Schindler, H. Moneva, J. van Pinxten, L. van Gool, B. van der Meulen, N. Stotz, and B. Theelen, “JetBrains MPS as core DSL technology for developing professional digital printers,” in *Domain-Specific Languages in Practice*, Springer, 2021, pp. 53–91.

[100] M. Simi, “Learning data analysis with MetaR,” in *Domain-Specific Languages in Practice*, Springer, 2021, pp. 259–290.

[101] N. Stotz and K. Birken, “Migrating insurance calculation rule descriptions from Word to MPS,” in *Domain-Specific Languages in Practice*, Springer, 2021, pp. 165–194.

[102] P. Munk and A. Nordmann, “Model-based safety assessment with SysML and component fault trees: Application and lessons learned,” *Software and Systems Modeling*, vol. 19, no. 4, pp. 889–910, 2020.

[103] A. Bucchiarone, M. Savary-Leblanc, X. L. Pallec, J.-M. Bruel, A. Cicchetti, J. Cabot, S. Gerard, H. Aslam, A. Marconi, and M. Perillo, “Papyrus for gamers, let’s play modeling,” in *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, 2020, pp. 1–5.

[104] M. V. Merino, J. Bartels, M. van den Brand, T. van der Storm, and E. Schindler, “Projecting textual languages,” in *Domain-Specific Languages in Practice*, Springer, 2021, pp. 197–225.

[105] R. Cuinat, C. Teodorov, and J. Champeau, “SpecEdit: Projectional editing for TLA+ specifications,” in *2020 IEEE Workshop on Formal Requirements (FORMREQ)*, IEEE, 2020, pp. 1–7.

[106] A. Prinz, “Teaching language engineering using MPS,” in *Domain-Specific Languages in Practice*, Springer, 2021, pp. 315–336.

[107] M. Barash and V. Pech, “Teaching MPS: Experiences from industry and academia,” in *Domain-Specific Languages in Practice*, Springer, 2021, pp. 293–313.

[108] B. Hempel and R. Chugh, “Tiny structure editors for low, low prices!(generating guis from tostring functions),” in *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, IEEE, 2020, pp. 1–5.

[109] E. Negm, S. Makady, and A. Salah, “Towards ontology-based domain specific language for internet of things,” in *Proceedings of the 2020 9th International Conference on Software and Information Engineering (ICSIE)*, 2020, pp. 146–151.

[110] J. Lubin and R. Chugh, “Type-directed program transformations for the working functional programmer,” in *10th Workshop on Evaluation and Usability of Programming Languages and Tools (PLATEAU 2019)*, 2020.

[111] M. Ozkaya and D. Akdur, “What do practitioners expect from the meta-modeling tools? a survey,” *Journal of Computer Languages*, 2021.

Appendix A

Protocol Validation Checklist

The protocol will be validated using the checklist in table A.1, which was adapted from Kitchenham's book Evidence-Based Software Engineering and Systematic Reviews[64].

Components	Questions
Background	Is the motivation for the review clearly stated and reasonable?
Research questions	Do these address a topic of interest to practitioners and/or researchers? Are they clearly stated?
Search strategy	Is the strategy justified and is it likely to find the right primary studies without the reviewers having to check or read a large number of irrelevant papers? Has the strategy been validated?
Study selection	Are the inclusion/exclusion criteria clearly defined and related to research questions? Is a validation process specified? Is there a process for handling marginal and uncertain papers? Is there a process for managing multiple reports of individual studies?
Quality of primary studies	Is it clear that the outcomes will be used in the later stages of the review? Is a validation process specified? Are criteria for assessing quality provided and justified and appropriate to the anticipated primary study types?
Data extraction	Does the data to be extracted properly address the research questions? Are the methods of recording the data appropriate for the types of data to be extracted? Is a validation process specified? Are there mechanisms for iteration where data is qualitative and categories are not (or cannot be) fully defined in advance of the extraction? Will the process enable the research questions to be answered?
Data aggregation and synthesis	Are the methods proposed for qualitative and quantitative data appropriate? Has consideration been given to combining results across multiple study types? Is the approach to aggregation and synthesis justified concerning appropriate literature?
Reporting	Has this been considered? has sufficient attention been paid to the completeness, general interest, validation, traceability, and the limitations of the review?
Review management	Are the tools that will be used for managing papers, studies, and data specified and appropriate? Is the management of the many-to-many relationship between papers and studies addressed?

TABLE A.1: Protocol validation checklist

Appendix B

Systematic Literature Review Log

Search Description

The papers we found with our automatic search can be found in table B.2. There were 100 unique papers found with the search strings across all of the venues when we used the search string and date restrictions. First we excluded those that were not in English or were unavailable to us. This reduced the count to 94. Next we downloaded each of the papers.

Next we skimmed all of these papers to remove any that were obviously unrelated to projectional editing. This reduced the count to 69.

two of the papers were referring to the same study, which reduced the count to 67.

We then excluded all grey literature, i.e. masters projects, proposals and PhD theses, and also books. This brought us down to 51 papers.

At this point we began our quality assessment.

Table description

Table B.2 shows the log of the Systematic literature review.

The First column "Paper Title" is the name of the paper as given by the search engine.

The second column "Lib", indicates the library or search engine through which it was found. The Libraries are Identified by the Keys in table B.1.

Key	Search engine/library	Key	Search engine/library
1	Google Scholar	2	IEEExplores
3	ACM	4	BASE
5	CORE	6	Web of Science
7	Microsoft Academic	8	SCOPUS
9	Semantic Scholar	10	SpringerLink
11	Wiley Online	12	Science.gov

TABLE B.1: Search engine/library key

The third and fourth columns show inclusion and exclusion reasons. As inclusions only rely on one question, “does this paper discuss projectional editing”, the affirmative is indicated by a tick. The exclusion column includes the reason for the exclusion.

The final column “ref”, gives a link to the citations in the separate bibliography for the systematic literature review, found at the end of the Appendices.

Paper Title	lib	in	exclusion	F#	B#
“Filmar, assistir e problematizar” – contribuições à aprendizagem de cálculos	9		not English	X	X
20. Internationales Stuttgarter Symposium	10		book	X	X
A Domain-Specific Language for Payroll Calculations: a Case Study at DATEV	1	✓		0	0
A Domain-Specific Language for Payroll Calculations: An Experience Report from DATEV	1,10	✓	Duplicate	X	X
A Framework for Modernizing Domain-Specific Languages	1	✓	grey	X	X
A Framework for Projectional Multi-variant Model Editors	1,8	✓		0	0
A Generic Projectional Editor for EMF Models	1,7,8,9	✓		2	0
A language-driven Development framework for simulation components to generate simulated environments	1	✓	grey	X	X
A Model-Driven Approach Towards Automatic Migration to Microservices	10	✓		5	0
A survey of Model Driven Engineering in robotics	1	✓		2	2
A Survey on the Design Space of End-User Oriented Languages for Specifying Robotic Missions	1,10	✓		1	5
A survey on the formalisation of system requirements and their validation	1	✓		0	0
A text-based syntax completion method using LR parsing	1			X	X
Activities and costs of re-engineering cloned variants into an integrated platform	1			X	X
AdaptiveVLE: An Integrated Framework for Personalized Online Education Using MPS JetBrains Domain-Specific Modeling Environment	1,2	✓		1	1
Adding Interactive Visual Syntax to Textual Code	3	✓		3	0
An approach to generate text-based IDEs for syntax completion based on syntax specification	1	✓		1	0
An MPS implementation for SimpliC	1	✓	grey	x	x
Blended graphical and textual modelling for UML profiles: A proof-of-concept implementation and experiment	1	✓		0	0
Block-based syntax from context-free grammars	1,3,5	✓		0	2
Bridging the worlds of textual and projectional language workbenches	1	✓	grey	X	X
Classification Algorithms Framework (CAF) to Enable Intelligent Systems Using JetBrains MPS Domain-Specific Languages Environment	2,5	✓		4	0

Paper Title	lib	in	exclusion	F#	B#
Code and Structure Editing for Teaching: A Case Study in using Bibliometrics to Guide Computer Science Research	1,9	✓		2	0
ComPOS - a Domain-Specific Language for Composing Internet-of-Things Systems	1,4	✓	grey	X	X
Concepts of variation control systems	1	✓		9	5
Concise, Type-Safe, and Efficient Structural Diffing	3			X	X
Constructing optimized constraint-preserving application conditions for model transformation rules	1			X	X
Design & Evaluation of an Accessible High-Level Language for Advanced Cryptography	1	✓	grey	X	X
Domain-specific languages for modeling and simulation	1			X	X
Domain-Specific Languages in Practice	10	✓	book	X	X
DSL Based Approach for Building Model-Driven Questionnaires	1,10	✓		0	1
DSS-Based Ontology Alignment in Solid Reference System Configuration	10		unavailable	X	X
Editing Software as Strategy Value	1			X	X
Efficient editing in a tree-oriented projectional editor	1,3,7,8,9	✓		1	0
Efficient generation of graphical model views via lazy model-to-text transformation	1,4	✓		1	0
Efficient usage of abstract scenarios for the development of highly-automated driving functions	1,10	✓	unavailable	X	X
Enabling language engineering for the masses	1,3	✓		2	1
Engineering Gameful Applications with MPS	1,10	✓		0	2
Enhancing development and consistency of UML models and model executions with USE studio	1,3,7,8,9	✓		0	2
Enterprise Information Systems	10		book	X	X
Example-driven software language engineering	1,3	✓		1	2
Exploring Visual Primitives for Authoring Source Code	1		grey	X	X
FASTEN: An Extensible Platform to Experiment with Rigorous Modeling of Safety-Critical Systems	1,10	✓		0	5
FeatureCoPP: unfolding preprocessor variability	1,3			X	X

Paper Title	lib	in	exclusion	F#	B#
FeatureVista: Interactive Feature Visualization	1			X	X
Filling Typed Holes with Live GUIs	3	✓		0	5
First-class concepts: reifying architectural knowledge beyond the dominant decomposition	1,3	✓		0	1
FORMREQ 2020	1,2,8		book	X	X
Gentleman: a light-weight web-based projectional editor generator	1,3,4,7,8,9	✓		0	0
GPP, the Generic Preprocessor	1			X	X
Improving the usability of the domain-specific language editors using artificial intelligence	1	✓	grey	X	X
Incremental Flow Analysis through Computational Dependency Reification	1,2	✓		0	2
Incrementalizing Static Analyses in Datalog	1	✓	grey	X	X
Integrating the Common Variability Language with Multilanguage Annotations for Web Engineering	1			X	X
Integrating UML and ALF: An Approach to Overcome the Code Generation Dilemma in Model-Driven Software Engineering	10	✓		0	0
Javardise: a structured code editor for programming pedagogy in Java	1	✓		0	0
JetBrains MPS as Core DSL Technology for Developing Professional Digital Printers	1,10	✓		0	0
JetBrains MPS: Why Modern Language Workbenches Matter	1,7,10	✓		0	1
Learning Data Analysis with MetaR	1,10	✓		0	0
Lipschitz-like property relative to a set and the generalized Mordukhovich criterion	6			X	X
Macros for Domain-Specific Languages	3			X	X
Mechanizing metatheory interactively	1	✓	grey	X	X
Migrating Insurance Calculation Rule Descriptions from Word to MPS	1,10	✓		0	0
Model Driven Software Engineering Meta-Workbenches: An XTools Approach	1,5	✓		0	0
Model-based safety assessment with SysML and component fault trees: application and lessons learned	1,10	✓		8	0
Model-Driven Development for Spring Boot Microservices	1,5	✓	grey	X	X
n Challenges for Software Language Engineering	1			X	X
On preserving variability consistency in multiple models	1,3			X	X

Paper Title	lib	in	exclusion	F#	B#
On the Need for a Formally Complete and Standardized Language Mapping between C++ and UML	1			X	X
On the Understandability of Language Constructs to Structure the State and Behavior in Abstract State Machine Specifications: A Controlled Experiment	1			X	X
On the use of product-line variants as experimental subjects for clone-and-own research: a case study	1			X	X
PAMOJA: A component framework for grammar-aware engineering	1	✓		0	1
Programming Robots for Activities of Everyday Life	1	✓	grey	X	X
Programming tools for intelligent systems	1	✓	grey	X	X
Projecting Textual Languages	1,10	✓		0	1
Rule-based and user feedback-driven decision support system for transforming automatically-generated alignments into information-integration alignments	5		unavailable	X	X
Semi-Automatische Deduktion von Feature-Lokalisierung während der Softwareentwicklung: Masterarbeit	5		not English	X	X
Should Variation Be Encoded Explicitly in Databases?	1			X	X
SLang: A Domain-specific Language for Survey Questionnaires	1	✓		0	0
SpecEdit: Projectional Editing for TLA+ Specifications	1,4,7	✓		0	0
Specifying Software Languages: Grammars, Projectional Editors, and Unconventional Approaches	1,2,4,5,7,8,9	✓		0	6
Teaching Language Engineering Using MPS	10	✓		0	0
Teaching MPS: Experiences from Industry and Academia	1,10	✓		0	1
Teasy framework: uma solução para testes automatizados em aplicações web	1	✓	not English	X	X
The Art of Bootstrapping	10	✓		3	0
The state of adoption and the challenges of systematic variability management in industry	1			X	X
Toward a domain-specific language for scientific workflow-based applications on multicloud system	1,11			X	X
Towards a Universal Variability Language	1	✓	grey	X	X

Paper Title	lib	in	exclusion	F#	B#
Towards Multi-editor Support for Domain-Specific Languages Utilizing the Language Server Protocol	10	✓		5	0
Towards Ontology-based Domain Specific Language for Internet of Things	1,3	✓		0	0
Towards projectional editing for model-based SPLs	3,4,7,8,9	✓		3	0
Tychonis: A model-based approach to define and search for geometric events in space	1			X	X
Type-Directed Program Transformations for the Working Functional Programmer	1	✓		0	0
Understanding Variability-Aware Analysis in Low-Maturity Variant-Rich Systems	1			X	X
Untangling Mechanized Proofs	3			X	X
Variability representations in class models: An empirical assessment	1,3			X	X
Visual design for a tree-oriented projectional editor	1,3,4,7,8,9	✓	Duplicate	X	X
What do practitioners expect from the meta-modeling tools? A survey	1,7,8,9	✓		0	3
Cyrillic named paper 1	1		not English	X	X
Cyrillic named paper 2	1		not English	X	X

TABLE B.2: Systematic review log - search results

Appendix C

Study Quality Assessment Checklist

The Center for Evidence-Based Management (CEBMa) supports the application of evidence-based practices to the field of management and leadership. They have a collection of checklists for assessing different types of studies. These checklists have been adapted from the pocket guide to critical appraisal[83]. We gave used these as the basis of our quality assessment checklists.

Critical Appraisal of a Case Study

#	Appraisal questions	Yes	Can't tell	No
1	Did the study address a focused question/issue?			
2	Is the research method (study design) appropriate for answering the research question?			
3	Are both the setting and the subject's representative concerning the population to which the findings will be referred?			
4	Is the researcher's perspective clearly described and taken into account?			
5	Are the methods for collecting data clearly described?			
6	Are the methods for analyzing the data likely to be valid and reliable? Are quality-control measures used?			
7	Was the analysis repeated by more than one researcher to ensure reliability?			
8	Are the results credible, and if so, are they relevant for practice?			
9	Are the conclusions drawn justified by the results?			
10	Are the findings of the study transferable to other settings?			

TABLE C.1: Case studies quality assessment checklist

Critical Appraisal of a Qualitative Study

#	Appraisal questions	Yes	Can't tell	No
1	Did the study address a focused question/issue?			
2	Is the research method (study design) appropriate for answering the research question?			
3	Was the context clearly described?			
4	How was the fieldwork undertaken? Was it described in detail? Are the methods for collecting data clearly described?			
5	Could the evidence (fieldwork notes, interview transcripts, recordings, documentary analysis, etc.) be inspected independently by others?			
6	Are the procedures for data analysis reliable and theoretically justified? Are quality-control measures used?			
7	Was the analysis repeated by more than one researcher to ensure reliability?			
8	Are the results credible, and if so, are they relevant for practice?			
9	Are the conclusions drawn justified by the results?			
10	Are the findings of the study transferable to other settings?			

TABLE C.2: Qualitative studies quality assessment checklist

Critical Appraisal of a Survey Study

#	Appraisal questions	Yes	Can't tell	No
1	Did the study address a focused question/issue?			
2	Is the research method (study design) appropriate for answering the research question?			
3	Is the method of selection of the subjects (employees, teams, divisions, organizations) clearly described?			
4	Could the way the sample was obtained introduce (selection) bias?			
5	Was the sample of subjects representative concerning the population to which the findings will be referred?			
6	Was the sample size based on pre-study considerations of statistical power?			
7	Was a satisfactory response rate achieved?			
8	Are the measurements (questionnaires) likely to be valid and reliable?			
9	Was the statistical significance assessed?			
10	Are confidence intervals given for the main results?			
11	Could there be confounding factors that haven't been accounted for?			
12	Are the findings of the study transferable to other settings?			

TABLE C.3: Survey studies quality assessment checklist

Critical Appraisal of a Cohort or Panel Study

#	Appraisal questions	Yes	Can't tell	No
1	Did the study address a focused question/issue?			
2	Is the research method (study design) appropriate for answering the research question?			
3	Were there enough subjects (employees, teams, divisions, organizations) in the study to establish that the findings did not occur by chance?			
4	Was the selection of the cohort/panel based on external, objective, and validated criteria?			
5	Was the cohort/panel representative of a defined population?			
6	Was the follow up of cases/subjects long enough?			
7	Were objective and unbiased outcome criteria used?			
8	Are objective and validated measurement methods used to measure the outcome?			
9	Is the size effect practically relevant?			
10	How precise is the estimate of the effect? Were confidence intervals given?			
11	Could there be confounding factors that haven't been accounted for?			
12	Are the findings of the study transferable to other settings?			

TABLE C.4: Cohort or panel studies quality assessment checklist

Appendix D

Study Quality Assessment Results

In this appendix we present the data for the findings of the Quality assessment stage of the SLR. The occurred after the initial search engine selections, the three snowballing iterations and the final deep read for classification.

Naturally, this table only reports on primary studies. Where a paper reports on more than one study the paper title appears multiple times, with the type of study in parenthesis. If we had multiple papers reporting on the same study, they have already been removed.

We separated the studies into their types. When the authors self reported a type, even if we were not in agreement with them we categorised these as such. The study types were survey, case study, action design research, and qualitative study. for the sake of table width we refer to action design research as ADR.

The question assessments are based on the assessment criteria presented in Appendix C. However these criteria do not have a checklist for Active Design Research. After much research we did not find an adequate checklist for ADR, so we used the case study checklist. After concluding the Quality assessment we had to conclude either that this checklist was not a valid interrogation of ADR studies, or that all 20 ADR studies were bad.

To score the studies, we arbitrarily decided to gave a +1 value for positive answers, 0 for don't knows and -1 for negative answers. We understand that this is a crude system. Note that, whilst most questions answered with “Yes” were considered positive, in the survey checklist, question “Could the way the sample was obtained introduce (selection)bias?”, the positively scored answer is “No”.

Name	Type	Score	1	2	3	4	5	6	7	8	9	10	11	12
A Domain-Specific Language for Payroll Calculations: a Case Study at DATEV[84]	Case Study	3	Y	Y	Y	?	N	?	N	?	Y	Y	-	-
A Framework for Projectional Multi-variant Model Editors[85]	ADR	-5	N	?	?	N	N	?	N	?	?	N	-	-
A Generic Projectional Editor for EMF Models[86]	ADR	-5	N	?	?	N	N	?	N	?	?	N	-	-
A Model-Driven Approach Towards Automatic Migration to Microservices[87]	ADR	-5	N	?	?	N	N	?	N	?	?	N	-	-
AdaptiveVLE: An Integrated Framework for Personalized Online Education Using MPS JetBrains Domain-Specific Modeling Environment[88]	ADR	3	N	?	Y	?	Y	Y	N	Y	Y	?	-	-
Adding Interactive Visual Syntax to Textual Code[89]	ADR	-5	N	?	?	N	N	?	N	?	?	N	-	-
Blended graphical and textual modelling for UML profiles: A proof-of-concept implementation and experiments[90]	Qualitative Study	8	Y	Y	Y	Y	?	Y	?	Y	Y	Y	-	-
Block-based syntax from context-free grammars[90]	Case Study	-3	N	?	?	N	N	?	N	?	Y	?	-	-
Classification Algorithms Framework (CAF) to Enable Intelligent Systems Using JetBrains MPS Domain-Specific Languages Environment[91]	ADR	3	N	?	Y	?	Y	Y	N	Y	Y	?	-	-
DSL Based Approach for Building Model-Driven Questionnaires (Action Research)[92]	ADR	-5	N	?	?	N	N	?	N	?	?	N	-	-
DSL Based Approach for Building Model-Driven Questionnaires (Qualitative Study1)[92]	Qualitative Study	-2	N	?	Y	N	?	?	N	?	?	?	-	-
DSL Based Approach for Building Model-Driven Questionnaires (Qualitative Study2)[92]	Qualitative Study	-2	N	?	Y	N	?	?	N	?	?	?	-	-
Efficient editing in a tree-oriented projectional editor[93]	ADR	-5	N	?	?	N	N	?	N	?	?	N	-	-
Efficient generation of graphical model views via lazy model-to-text transformation[94]	ADR	0	N	?	?	N	Y	Y	N	Y	?	?	-	-
Engineering Gameful Applications with MPS[95]	ADR	-6	N	?	N	N	N	?	N	?	?	N	-	-
FASTEN: An Extensible Platform to Experiment with Rigorous Modeling of Safety-Critical Systems[96]	ADR	-4	N	?	N	?	N	?	N	?	?	?	-	-

Name	Type	Score	1	2	3	4	5	6	7	8	9	10	11	12
Gentleman: a light-weight web-based projectional editor generator[25]	ADR	-5	N	?	N	?	N	?	N	?	?	N	-	-
Integrating UML and ALF: An Approach to Overcome the Code Generation Dilemma in Model-Driven Software Engineering[97]	ADR	-5	N	?	N	?	N	?	N	?	?	N	-	-
Javardise: a structured code editor for programming pedagogy in Java[98]	ADR	-5	N	?	N	?	N	?	N	?	?	N	-	-
JetBrains MPS as Core DSL Technology for Developing Professional Digital Printers[99]	Case Study	-6	N	?	N	N	N	?	N	?	?	N	-	-
Learning Data Analysis with MetaR[100]	ADR	-4	N	?	Y	N	N	?	N	?	?	N	-	-
Migrating Insurance Calculation Rule Descriptions from Word to MPS[101]	Case Study	-3	N	?	Y	N	N	?	N	?	?	?	-	-
Model-based safety assessment with SysML and component fault trees: application and lessons learned (Case study1)[102]	Case Study	-4	Y	?	N	N	N	?	N	?	?	N	-	-
Model-based safety assessment with SysML and component fault trees: application and lessons learned (Case study2)[102]	Case Study	-2	Y	?	Y	N	N	?	N	?	?	N	-	-
Model-based safety assessment with SysML and component fault trees: application and lessons learned (Action Research)[102]	ADR	-3	N	?	Y	N	N	?	N	?	?	?	-	-
Papyrus for gamers, let's play modeling[103]	ADR	-4	N	?	?	N	N	?	N	?	?	?	-	-
Projecting Textual Languages (Action Research)[104]	ADR	-5	N	?	N	N	N	?	N	?	?	?	-	-
Projecting Textual Languages (Case Study)[104]	Case Study	-6	N	?	N	N	N	?	N	?	?	N	-	-
SpecEdit: Projectional Editing for TLA+ Specifications (Action Research)[105]	ADR	-2	Y	?	?	N	N	?	N	?	?	?	-	-
SpecEdit: Projectional Editing for TLA+ Specifications (Case Study)[105]	Case Study	-5	N	?	N	N	N	?	N	?	?	?	-	-
Teaching Language Engineering Using MPS[106]	Case Study	3	N	?	Y	Y	N	?	N	Y	Y	Y	-	-
Teaching MPS: Experiences from Industry and Academia[107]	Case Study	0	N	?	Y	Y	N	Y	N	?	?	?	-	-
Tiny Structure Editors for Low, Low Prices (Action Research)[108]	ADR	-5	N	?	?	N	N	?	N	?	?	N	-	-
Tiny Structure Editors for Low, Low Prices (Case Study)[108]	Case Study	-6	N	?	N	N	N	?	N	?	?	N	-	-

Name	Type	Score	1	2	3	4	5	6	7	8	9	10	11	12
Towards Ontology-based Domain Specific Language for Internet of Things[109]	ADR	-6	N	?	?	N	N	?	N	?	?	N	-	-
Type-Directed Program Transformations for the Working Functional Programmer[110]	ADR	-3	Y	Y	N	N	N	?	N	?	?	N	-	-
What do practitioners expect from the meta-modeling tools? A survey[111]	Survey	1	Y	Y	Y	Y*	?	?	Y	Y	N	N	?	N

TABLE D.1: Quality assessment results

Appendix E

Data Extraction Results

Study ID	1
Title of Study	A domain-specific language for payroll calculations: A case study at DATEV
Year of Publication	2021
Author(s) Names	M. Voelter, S. Koščejev, M. Riedel, A. Deitsch, and A. Hinkelmann
Source of Study	Google Scholar, SpringerLink
Type of Study	Case Study
Name of Venue	Domain-Specific Languages in Practice
Tools in Study	Jetbrains MPS
Sentiment	generallearnings -- negative = 23, neutral = 28, positive = 13 evaluation -- negative = 45, neutral = 69, positive = 23 conclusion -- negative = 3, neutral = 10, positive = 4

Study ID	2
Title of Study	A framework for projectional multi-variant model editors
Year of Publication	2021
Author(s) Names	J. Schröpfer, T. Buchmann, and B. Westfecht
Source of Study	Google Scholar, SCOPUS
Type of Study	Action Design Research
Name of Venue	MODELSWARD
Tools in Study	EMF & Ecore
Sentiment	intro -- negative = 2, neutral = 23, positive = 2 conclusion -- negative = 2, neutral = 9, positive = 3

Study ID	3
Title of Study	A generic projectional editor for EMF models
Year of Publication	2020
Author(s) Names	J. Schröpfer, T. Buchmann, and B. Westfecht
Source of Study	Google Scholar, Microsoft Academic, SCOPUS, Semantic Scholar
Type of Study	Action Design Research
Name of Venue	MODELSWARD
Tools in Study	EMF & Ecore
Sentiment	intro -- negative = 12, neutral = 43, positive = 8 conclusion -- negative = 2, neutral = 6, positive = 1

Study ID	4
Title of Study	A model-driven approach towards automatic migration to microservices
Year of Publication	2020
Author(s) Names	A. Bucchiarone, K. Soysal, and C. Guidi
Source of Study	SpringerLink
Type of Study	Action Design Research
Name of Venue	International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment
Tools in Study	Jetbrains MPS
Sentiment	intro -- negative = 1, neutral = 15, positive = 3 conclusion -- neutral = 4

FIGURE E.1: Data extraction results 1 - 4

Study ID	5
Title of Study	AdaptiveVLE: An integrated framework for personalized online education using MPS JetBrains domain-specific modeling environment
Year of Publication	2020
Author(s) Names	S. Meacham, V. Pech, and D. Nauck
Source of Study	Google Scholar, IEEExplores
Type of Study	Action Design Research
Name of Venue	IEEE Access
Tools in Study	Jetbrains MPS
Sentiment	intro -- negative = 3, neutral = 17, positive = 2 conclusion -- negative = 2, neutral = 3, positive = 3

Study ID	6
Title of Study	Adding interactive visual syntax to textual code
Year of Publication	2020
Author(s) Names	L. Andersen, M. Ballantyne, and M. Felleisen
Source of Study	ACM
Type of Study	Action Design Research
Name of Venue	Proceedings of the ACM on Programming Languages OOPSLA
Tools in Study	Racket and DrRacket
Sentiment	conclusion -- negative = 1, neutral = 14, positive = 4

Study ID	7
Title of Study	Blended graphical and textual modelling for UML profiles: A proof-of-concept implementation and experiment
Year of Publication	2021
Author(s) Names	L. Addazi and F. Ciccozzi
Source of Study	Google Scholar
Type of Study	Qualitative Study
Name of Venue	Journal of Systems and Software
Tools in Study	Xtext & Ecore
Sentiment	intro -- negative = 6, neutral = 21, positive = 5 projectionalediting -- neutral = 6 discussion -- negative = 18, neutral = 26, positive = 6 conclusion -- negative = 1, neutral = 5, positive = 4

Study ID	8
Title of Study	Classification algorithms framework (CAF) to enable intelligent systems using JetBrains MPS domain-specific languages environment
Year of Publication	2020
Author(s) Names	S. Meacham, V. Pech, and D. Nauck
Source of Study	IEEExplores, CORE
Type of Study	Action Design Research
Name of Venue	IEEE Access
Tools in Study	Jetbrains MPS
Sentiment	intro -- negative = 2, neutral = 19 evaluation -- negative = 3, neutral = 10, positive = 10 conclusion -- negative = 1, neutral = 6

Study ID	9
Title of Study	DSL based approach for building model-driven questionnaires
Year of Publication	2020
Author(s) Names	A. L. Furtado
Source of Study	Google Scholar, SpringerLink
Type of Study	Action Design Research + Qualitative Study
Name of Venue	Enterprise Information Systems: 22nd International Conference ICEIS 2020
Tools in Study	Jetbrains MPS
Sentiment	intro -- negative = 4, neutral = 11, positive = 3 Implementations -- negative = 4, neutral = 16 conclusion -- negative = 3, neutral = 5, positive = 2

FIGURE E.2: Data extraction results 5 - 9

Study ID	10
Title of Study	Efficient editing in a tree-oriented projectional editor
Year of Publication	2020
Author(s) Names	T. Beckmann
Source of Study	Google Scholar, ACM, Microsoft Academic,SCOPUS, Semantic Scholar
Type of Study	Action Design Research
Name of Venue	Conference Companion of the 4th International Conference on Art, Science, and Engineering of Programming
Tools in Study	sandblocks (Research Project)
Sentiment	intro -- negative = 1, neutral = 7, positive = 2 design -- neutral = 9 conclusion -- negative = 1, neutral = 1, positive = 1

Study ID	11
Title of Study	Efficient generation of graphical modelviews via lazy model-to-text transformation
Year of Publication	2020
Author(s) Names	D. Kolovos, A. De La Vega, and J. Cooper
Source of Study	Google Scholar, BASE
Type of Study	Action Design Research
Name of Venue	Proceedings of the 23rd ACM/IEEEInternational Conference on Model Driven Engineering Languages and Systems
Tools in Study	Picto (Research Project using EGL, Graphviz,PlantUML)
Sentiment	intro -- negative = 3, neutral = 5, positive = 1 evaluation -- negative = 5, neutral = 43, positive = 1 results -- negative = 5, neutral = 34, positive = 5 conclusion -- negative = 1, neutral = 3, positive = 3

Study ID	13
Title of Study	Engineering gameful applications with MPS
Year of Publication	2021
Author(s) Names	A. Bucchiarone, A. Cicchetti, and A. Marconi
Source of Study	Google Scholar, SpringerLink
Type of Study	Action Design Research
Name of Venue	Domain-Specific Languages in Practice
Tools in Study	Jetbrains MPS
Sentiment	intro -- negative = 6, neutral = 20, positive = 5 engineering -- negative = 2, neutral = 34, positive = 7 mpsprojectional -- negative = 2, neutral = 18 lessonslearned -- negative = 11, neutral = 16, positive = 5 conclusion -- negative = 1, neutral = 5, positive = 2

Study ID	15
Title of Study	Fasten: An extensible platform to experiment with rigorous modeling of safety-critical systems
Year of Publication	2021
Author(s) Names	D. Ratiu, A. Nordmann, P. Munk, C. Carlan, and M. Voelter
Source of Study	Google Scholar, SpringerLink
Type of Study	Action Design Research
Name of Venue	Domain-Specific Languages in Practice
Tools in Study	Jetbrains MPS
Sentiment	intro -- negative = 18, neutral = 40, positive = 11 platform -- neutral = 13, positive = 2 discussion -- negative = 3, neutral = 21, positive = 2 discussionMPS -- negative = 14, neutral = 25, positive = 8 conclusion -- negative = 3, neutral = 11, positive = 4

Study ID	16
Title of Study	Gentleman: A light-weight web-based projectional editor generator
Year of Publication	2020
Author(s) Names	L.E. Lafontant and E. Syriani
Source of Study	Google Scholar, ACM, BASE, Microsoft Academic, SCOPUS, Semantic Scholar
Type of Study	Action Design Research
Name of Venue	Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings
Tools in Study	Gentleman (Research Project)
Sentiment	intro -- negative = 8, neutral = 12, positive = 2 editor -- negative = 5, neutral = 43, positive = 5 implementation -- neutral = 10, positive = 1 projections -- neutral = 40, positive = 6 conclusion -- neutral = 4, positive = 2

FIGURE E.3: Data extraction results 10 - 16

Study ID	17
Title of Study	Integrating UML and ALF: An approach to overcome the code generation dilemma in model-driven software engineering
Year of Publication	2020
Author(s) Names	J. Schröpfer and T. Buchmann
Source of Study	SpringerLink
Type of Study	Action Design Research
Name of Venue	International Conference on Model-Driven Engineering and Software Development
Tools in Study	ALF (Xtext) + Valkyrie (GMF)
Sentiment	intro -- negative = 4, neutral = 24, positive = 1 discussion -- negative = 4, neutral = 12, positive = 5 UI -- neutral = 9, positive = 2 conclusion -- neutral = 5

Study ID	18
Title of Study	Javardise: A structured code editor for programming pedagogy in Java
Year of Publication	2020
Author(s) Names	A. L. Santos
Source of Study	Google Scholar
Type of Study	Action Design Research
Name of Venue	Conference Companion of the 4th International Conference on Art, Science,and Engineering of Programming
Tools in Study	Javardise (Research Project)
Sentiment	discussion -- negative = 16, neutral = 13, positive = 4 intro -- negative = 11, neutral = 11, positive = 1

Study ID	19
Title of Study	Jetbrains MPS as core DSL technology for developing professional digital printers
Year of Publication	2021
Author(s) Names	E. Schindler, H. Moneva, J. van Pinxten, L. van Gool, B. van der Meulen, N. Stotz,and B. Theelen
Source of Study	Google Scholar, SpringerLink
Type of Study	Case Study
Name of Venue	Domain-Specific Languages in Practice
Tools in Study	Jetbrains MPS
Sentiment	CollaborativeDSM -- negative = 5, neutral = 29, positive = 13 conclusion -- negative = 2, neutral = 22, positive = 2 intro -- negative = 1, neutral = 21, positive = 3 MPS -- negative = 9, neutral = 9, positive = 3

Study ID	20
Title of Study	Learning data analysis with metaR
Year of Publication	2021
Author(s) Names	M. Simi
Source of Study	Google Scholar, SpringerLink
Type of Study	Action Design Research
Name of Venue	Domain-Specific Languages in Practice
Tools in Study	Jetbrains MPS
Sentiment	intro -- negative = 1, neutral = 4, positive = 1 languagecompsition -- neutral = 22, positive = 3 MPS -- negative = 9, neutral = 56, positive = 6 conclusion -- negative = 2, neutral = 4, positive = 5

Study ID	21
Title of Study	Migrating insurance calculation rule descriptions from Word to MPS
Year of Publication	2021
Author(s) Names	N. Stotz and K. Birken
Source of Study	Google Scholar, SpringerLink
Type of Study	Case Study
Name of Venue	Domain-Specific Languages in Practice
Tools in Study	Jetbrains MPS
Sentiment	intro -- neutral = 5, positive = 1 solutiontechnology -- negative = 20, neutral = 26, positive = 11 evaluation -- negative = 37, neutral = 49, positive = 10 conclusion -- negative = 9, neutral = 8, positive = 7

FIGURE E.4: Data extraction results 17 - 21

Study ID	22
Title of Study	Model-based safety assessment with sysml and component fault trees: Application and lessons learned
Year of Publication	2020
Author(s) Names	P. Munk and A. Nordmann
Source of Study	Google Scholar, SpringerLink
Type of Study	Action Design Research + Case Study
Name of Venue	Software and Systems Modeling
Tools in Study	Jetbrains MPS
Sentiment	intro -- negative = 8, neutral = 11, positive = 6 realisation -- negative = 29, neutral = 46, positive = 11 discussion -- negative = 6, neutral = 4, positive = 2 conclusion -- negative = 4, neutral = 3, positive = 1

Study ID	23
Title of Study	Papyrus for gamers, let's play modeling
Year of Publication	2020
Author(s) Names	A. Bucchiarone, M. Savary-Leblanc, X. L. Pallec, J.M. Bruel, A. Cicchetti, J. Cabot,S. Gerard, H. Aslam, A. Marconi, and M. Perillo
Source of Study	snowball
Type of Study	Action Design Research
Name of Venue	Proceedings of the 23rd ACM/IEEE International Conference on Model-Driven Engineering Languages and Systems: Companion Proceedings
Tools in Study	Jetbrains MPS
Sentiment	intro -- negative = 5, neutral = 8, positive = 3

Study ID	24
Title of Study	Projecting textual languages
Year of Publication	2021
Author(s) Names	M. V. Merino, J. Bartels, M. van den Brand, T. van der Storm, and E. Schindler
Source of Study	Google Scholar, SpringerLink
Type of Study	Action Design Research + Case Study
Name of Venue	Domain-Specific Languages in Practice
Tools in Study	Jetbrains MPS + Rascal
Sentiment	intro -- negative = 9, neutral = 17, positive = 3 projectionalApproach-- negative = 5, neutral = 128, positive = 4 projectionalSyntax -- negative = 4, neutral = 42 limitation -- negative = 10, neutral = 25, positive = 6 discussion -- negative = 6, neutral = 16, positive = 9 conclusion -- negative = 4, neutral = 14, positive = 5

Study ID	25
Title of Study	SpecEdit: Projectional editing for TLA+ specifications
Year of Publication	2020
Author(s) Names	R. Cuinat, C. Teodorov, and J. Champeau
Source of Study	Google Scholar, BASE,Microsoft Academic
Type of Study	Action Design Research + Case Study
Name of Venue	2020 IEEE Workshop on Formal Requirements (FORMREQ)
Tools in Study	Jetbrains MPS
Sentiment	intro -- negative = 4, neutral = 16, positive = 6 projectionalEditor -- negative = 7, neutral = 78, positive = 17 lessonslearned -- negative = 4, neutral = 6, positive = 4 discussion -- neutral = 5, positive = 1

Study ID	26
Title of Study	Teaching language engineering using MPS
Year of Publication	2021
Author(s) Names	A. Prinz
Source of Study	SpringerLink
Type of Study	Case Study
Name of Venue	Domain-Specific Languages in Practice
Tools in Study	Jetbrains MPS
Sentiment	intro -- negative = 6, neutral = 25, positive = 8 lessonslearned -- negative = 2, neutral = 12, positive = 2 metalanguages -- negative = 11, neutral = 25, positive = 11 MPSinTeaching -- negative = 14, neutral = 38, positive = 9 selectingTools -- negative = 6, neutral = 11, positive = 6 evaluation -- negative = 9, neutral = 30, positive = 20 conclusion -- negative = 2, neutral = 3, positive = 3

FIGURE E.5: Data extraction results 22 - 26

Study ID	27
Title of Study	Teaching MPS: Experiences from industry and academia
Year of Publication	2021
Author(s) Names	M. Barash and V. Pech
Source of Study	Google Scholar, SpringerLink
Type of Study	Case Study
Name of Venue	Domain-Specific Languages in Practice
Tools in Study	Jetbrains MPS
Sentiment	explainingprojectional -- negative = 1, neutral = 6, positive = 1 conclusion -- negative = 1, neutral = 5, positive = 1

Study ID	28
Title of Study	Tiny structure editors for low, low prices! (generating guis from toString functions)
Year of Publication	2020
Author(s) Names	B. Hempel and R. Chugh
Source of Study	snowball
Type of Study	Action Design Research + Case Study
Name of Venue	2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)
Tools in Study	Tiny Structure Editor (Research Project)
Sentiment	intro -- negative = 7, neutral = 30, positive = 2 discussion -- negative = 5, neutral = 11, positive = 1

Study ID	29
Title of Study	Towards ontology-based domain specific language for internet of things
Year of Publication	2020
Author(s) Names	E. Negm, S. Makady, and A. Salah
Source of Study	Google Scholar, ACM
Type of Study	Action Design Research
Name of Venue	Proceedings of the 2020 9th International Conference on Software and Information Engineering (ICSIE)
Tools in Study	Jetbrains MPS
Sentiment	intro -- negative = 8, neutral = 22, positive = 5 implementation -- neutral = 20, positive = 2 conclusion -- neutral = 7

Study ID	30
Title of Study	Type-directed program transformations for the working functional programmer
Year of Publication	2020
Author(s) Names	J. Lubin and R. Chugh
Source of Study	Google Scholar
Type of Study	Action Design Research
Name of Venue	10th Workshop on Evaluation and Usability of Programming Languages and Tools (PLATEAU 2019)
Tools in Study	DEUCE
Sentiment	intro -- negative = 7, neutral = 9, positive = 2 implementation -- negative = 13, neutral = 16, positive = 6 usabilityChallenges -- negative = 3, neutral = 5, positive = 2

Study ID	31
Title of Study	What do practitioners expect from the meta-modeling tools? a survey
Year of Publication	2021
Author(s) Names	M. Ozkaya and D. Akdur
Source of Study	Google Scholar, Microsoft Academic, SCOPUS, Semantic Scholar
Type of Study	Survey
Name of Venue	Journal of Computer Languages
Tools in Study	MPS, MetaEdit+, WebGME, GEMS, sirius,Xtext, MS DSL ToolsMelange, GME*
Sentiment	intro -- negative = 11, neutral = 62, positive = 8 editorservice -- negative = 8, neutral = 20, positive = 6 lessonslearned -- negative = 10, neutral = 24, positive = 8 challenges -- negative = 5 toolusage -- negative = 1, neutral = 14, positive = 2 conclusion -- negative = 10, neutral = 20, positive = 3

FIGURE E.6: Data extraction results 27 - 31

Appendix F

Drools Concept hierarchy

The concept hierarchy presented on the following pages was extracted and interpreted from Drools railroad diagrams.

The diagram in figure F.1 represents the file level and can be considered the root of concept hsierarchy. This represents the concepts that are available to the rule file. As the only concept we will examine in depth is the rule, we show some concepts that are shared or are children of, for example, function, query and type declaration.

In our final implementation only the import, global and Rule concepts that were children of the rule file were implemented.

The diagram in figure F.2 shows the children of a rule. Each attribute has a different behavior and structure and are thus all represented separately.

In These diagrams we do not show a concept diagram for the RHS. This is because it would be more or less the concept diagram for Java Statements,with the addition of Rule Variables and some special Drools functions. the concept diagram for a General Purpose Language, like Java, would be orders of magnitude bigger and more complex. Luckily, as MPS allows for almost seamless extension and integration of different languages, we are able to just import JetBrains implementation of Java for the RHS.

The hierarch for the LHS is shown in the diagram in figure F.3. Because of the number of concepts being represented, it may be a little hard to read.

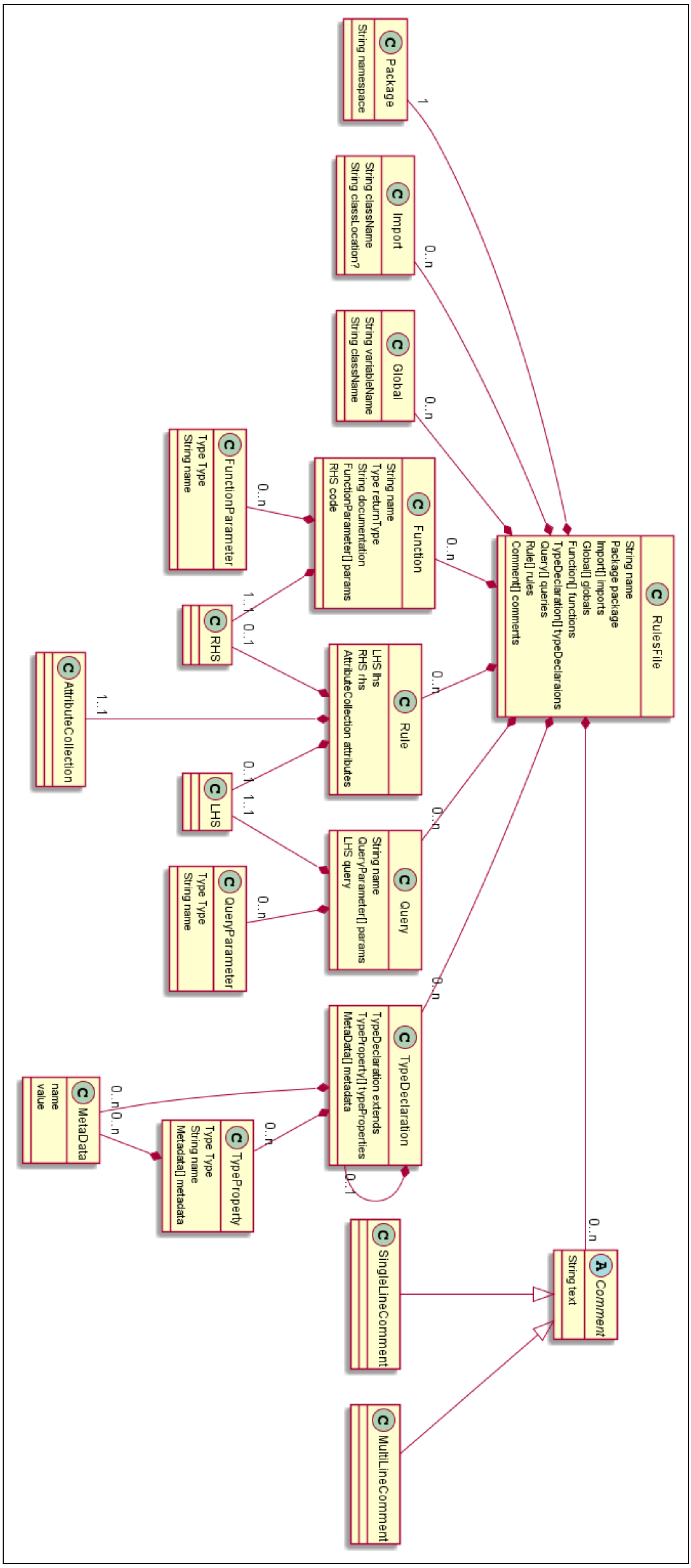


FIGURE F.1: Rule file concept hierarchy diagram

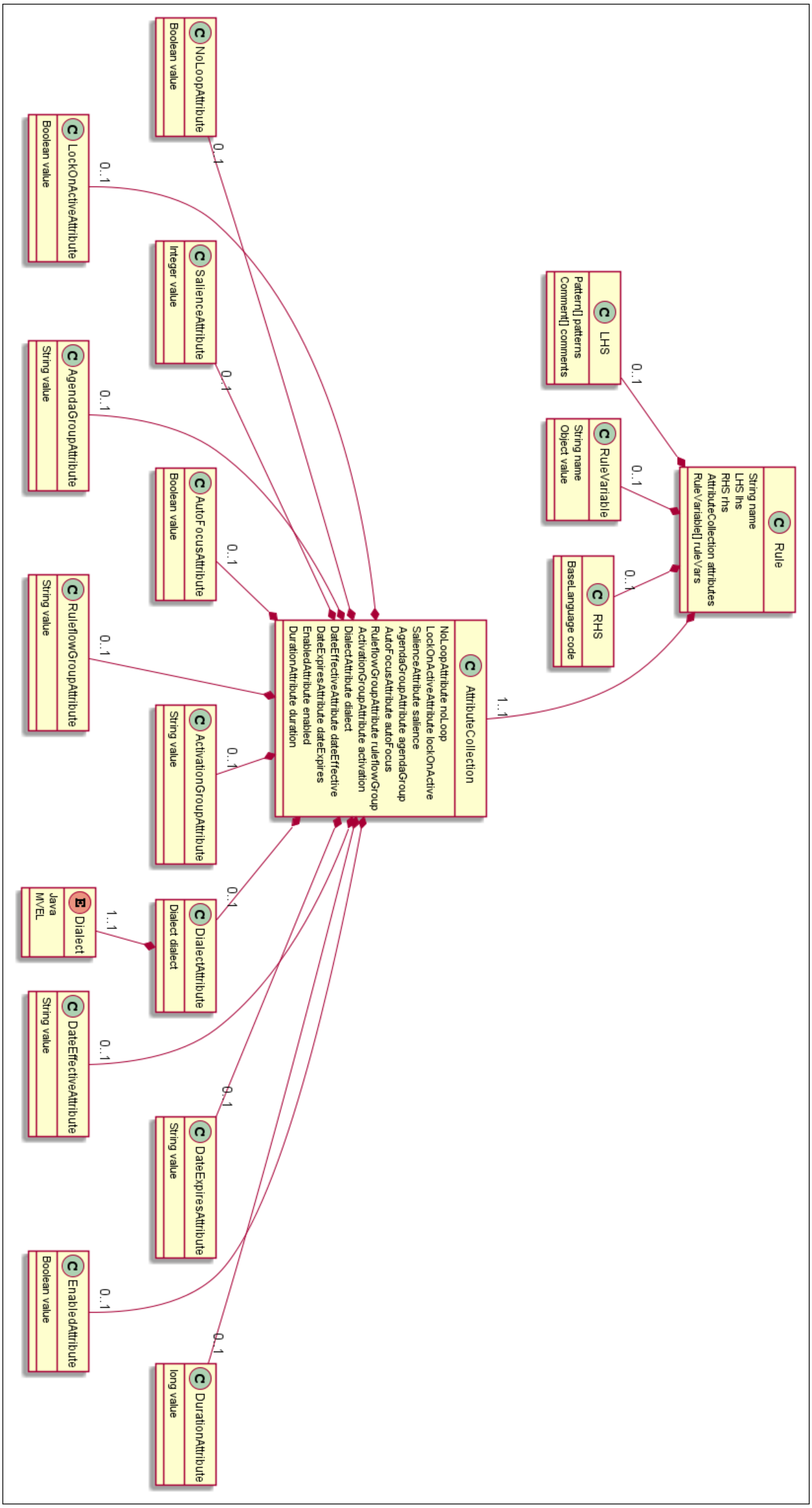


FIGURE F.2: Rules concept hierarchy diagram

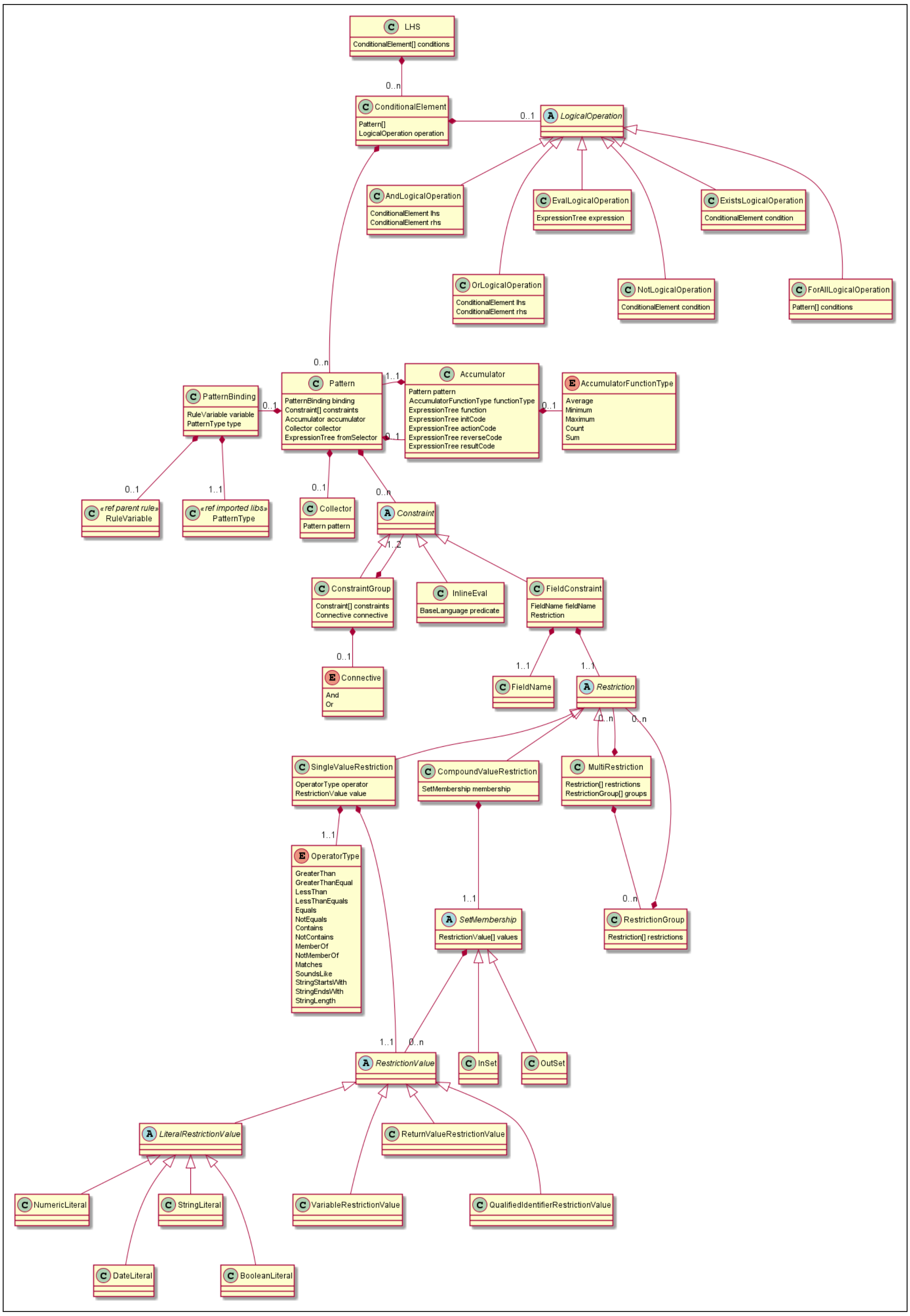


FIGURE F.3: Rule LHS concept hierarchy diagram

Appendix G

Questionnaire Text

Note: This questionnaire was presented on Survey Monkey and thus the text here is a best approximation of their paging system.

Page 1 - Introduction

Thank you for taking part in this research.

According to Survey Monkey, this survey should take 6 minutes to complete, when we tested it, the average was closer to 10 minutes.

This survey is for the validation section of a research master's project by Paul Spencer at the University of Amsterdam.

The purpose is to determine whether projectional editing can be used to aid the comprehensibility of business rules.

We are using Drools as our example business rules language.

You were selected as you asked or answered a Drools question on StackOverflow, listed Drools as a skill on your LinkedIn profile, or were referred to this survey by someone who previously answered this survey. (please feel free to forward this survey to anyone you know with Drools experience).

It is therefore assumed you are aware of what Drools is.

Projectional editing is a form of writing computer programs directly rather than writing text and having that parsed to create the program. This allows the developer multiple views and editors for the same code.

In this survey, we will present you with a few of these views.

On the following page, there is an animated GIF that will give a small demonstration of what this means.

Figure G.1 shows how this is presented to the subject.

Page 2 - Example of Projectional editing in Drools

Below is an animated GIF showing an example of a projectional implementation of Drools.

The top section is a tabular projection of the program.

The bottom part is a textual projection of the same program shown at the same time.

In this recording, we are editing in the tabular projection, which automatically updates the textual projection.

Here is placed an animated GIF of a demonstration of our prototype

Question: What is your first reaction to this mode of code editing?

Options: Very positive, Somewhat positive, Neutral, Somewhat negative, Very negative

the order of the options will be randomly presented as either "Very positive" to "Very negative" or "Very neg

Figure G.2 shows how this is presented to the subject.

Page 3 - Positive about projectional editing

This page is only selected if the user chose very positive or somewhat positive

This question is optional.

you may use the Green "PREV" button to review the previous page.

Question: how would this coding style be useful to your interactions with Drools?

This is an open question with a text box.

Figure G.3 shows how this is presented to the subject.

Page 4 - negative about projections

This page is only selected if the user chose very positive or somewhat positive

This question is optional.

you may use the Green "PREV" button to review the previous page.

Question: What do you find negative with this style of coding

This is an open question with a text box.

Page 5 - Testing a projection

In questionnaire version A & D page 5 will be Testing a projection

In questionnaire version B & C page 5 will be Testing textual projection

On this page, we present you with an example projection of a collection of Drools rules, in this case, as a sort of decision table.

We will ask you to describe what you think it does, if you can't that is also good data for us.

A brief description of how this projection works follows:

for the decision table the following text:

1) each row is a rule

2) each column is a fact, or, when indented, a selection criteria of that fact

3) smiley faces indicate that a fact has been selected for a rule

4) if a fact has been selected and a variable is bound to it then the variable name appears instead of the smiley face.

5) the "Then" part of the rule appears in the "Actions" column

for the other table the following text:

1) each row is a rule

2) each column is for a variable or a property of a fact

3) if a property is selected then the selection criteria is in the appropriate cell

4) unselected cells are indicated by a grey/beige color

5) the "Then" part of the rule appears in the "Actions" column

depending on the version of this questionnaire the respondent will see one of the following pictures

Version A - decision table showing rule set 1 (FNWI)

Version B - decision table showing rule set 2 (LAW)

Version C - new table showing rule set 1

Version D - new table showing rule set 2

Question: Please describe what you think this group of rules does

This is an open question with a text box.

Question: How easy or difficult was it to describe this rule set?

Options: Very easy, Somewhat easy, Neutral, Somewhat difficult, Very difficult

the order of the options will be randomly presented as either "Very easy" to "Very difficult" or "Very difficult" to "

Figure G.4 shows how this is presented to the subject.

Page 6 - Testing textual projection

In questionnaire version A & D page 6 will be Testing textual projection

In questionnaire version B & C page 6 will be Testing a projection

Here we present you a textual projection of Drools rules.

[Note: These are not the same rules as on the previous page]

depending on the version of this questionnaire the respondent will see one of the following pictures

Version A & C - a text projection of rule set 2 (LAW)

Version B & D - a text projection of rule set 1 (FNWI)

Question: Please describe what you think this group of rules does

This is an open question with a text box.

Question: How easy or difficult was it to describe this rule set?

Options: Very easy, Somewhat easy, Neutral, Somewhat difficult, Very difficult

the order of the options will be randomly presented as either “Very easy” to “Very difficult” or “Very difficult

Figure G.5 shows how this is presented to the subject.

Page 7 - Comparing projections 1

In this question, we ask to compare a new projection to a previously shown projection, on the page named “Testing a projection”.

If you wish to reacquaint yourself with the previous projection, you can use the Green “PREV” button at the bottom of this page.

A brief description of how this new projection works follows:

for the decision table the following text:

1) each row is a rule

2) each column is a fact, or, when indented, a selection criteria of that fact

3) smiley faces indicate that a fact has been selected for a rule

4) if a fact has been selected and a variable is bound to it then the variable name appears instead of the smiley face.

5) the “Then” part of the rule appears in the “Actions” column

for the other table the following text:

1) each row is a rule

2) each column is for a variable or a property of a fact

3) if a property is selected then the selection criteria is in the appropriate cell

4) unselected cells are indicated by a grey/beige color

5) the “Then” part of the rule appears in the “Actions” column

depending on the version of this questionnaire the respondent will see one of the following pictures

Version A - new table showing rule set 1

Version B - new table showing rule set 2

Version C - decision table showing rule set 1

Version D - decision table showing rule set 2

Question: How does the above projection compare to the first projection you described?

Options: Much easier to understand, Somewhat easier to understand, Neutral, Somewhat harder to understand, Much harder to understand

the order of the options will be randomly presented as either “Much easier to understand” to “Much harder to un

Figure G.6 shows how this is presented to the subject.

Page 8 - Comparing projections 2

In this question, we again ask to compare the new projection, this time to the textual projection, on the page named “Testing textual projection”.

If you wish to reacquaint yourself with the textual projection, you can, of course, use the Green “PREV” button at the bottom of this page again.

depending on the version of this questionnaire the respondent will see one of the following pictures

Version A - new table showing rule set 2

Version B - new table showing rule set 1

Version C - decision table showing rule set 2

Version D - decision table showing rule set 1

Question: How does the above projection compare to the text Drools rules you described?

Options: Much easier to understand, Somewhat easier to understand, Neutral, Somewhat harder to understand, Much harder to understand

the order of the options will be randomly presented as either “Much easier to understand” to “Much harder to un

Figure G.7 shows how this is presented to the subject.

Page 9 - Single rule helper 1 - Truth table

In questionnaire version A & D page 9 will be the Truth Table

In questionnaire version B & C page 9 will be the Circuit Diagram

Below we present another projection. This is a truth table projection. It highlights the conditions that have to be true for a rule to be selected.

The GIF shows the rule selected and the developer pressing the up and down arrow keys to step through the different true (highlighted in green) and false (highlighted in red) fact selections that result in a true outcome.

An animated GIF of the truth table example

Question: Would this help you with understanding your Drools rules?

Options: It would really help understanding, it would somewhat help understanding, Neutral, It would add a little confusion, It would add a lot of confusion

the order of the options will be randomly presented as either “It would really help understanding” to “It wo

Figure G.8 shows how this is presented to the subject.

Page 10 - Single rule helper 2 - Circuit Diagram

In questionnaire version A & D page 10 will be the Circuit Diagram

In questionnaire version B & C page 10 will be the Truth Table

This is a circuit diagram of the selection conditions. choosing a different condition highlights how they are related to each other.

The GIF shows the rule selected and the developer pressing the up and down arrow keys to step through the different fact selections (highlighted in yellow) and shown in the circuit diagram, thus showing how the facts relate to each other.

An animated GIF of the Circuit Diagram example

Question: Would this help you with understanding your Drools rules?

Options: It would really help understanding, it would somewhat help understanding, Neutral, It would add a little confusion, It would add a lot of confusion

the order of the options will be randomly presented as either “It would really help understanding” to “It wo

Figure G.9 shows how this is presented to the subject.

Page 11 - The Statistics page

Here we ask for data that we can use to slice and dice results.

Question: How long was/is your career as a developer?

Options: 0-1 year, 1-3 years, 3-10 years, greater than 10 years, none of the above

Question: When was the last time you had a coding interaction with Drools?

Options: during this week, some time after July 1st 2021, some time after Jan 1st 2021, some time after 2016, some time before 2016

Question: how long did you work with Drools?

Options: for years and intensely, for years but occasionally, not for long but intensely, I barely touched it

Question: Which tools have you used to edit Drools rules?

Checkboxes: Drools workbench, eclipse (with drools plugin), IntelliJ IDEA (with drools plugin), IDE or text editor without Drools assistance, other (please specify) *has textbox*, none of the above

Figure G.10 shows how this is presented to the subject.

Page 12 - So long, and thanks for all the fish

Thank you for your time. We leave you with a box where you can put in any thoughts about this if you feel like it.

Question: Do you have any thoughts or opinions you would like to share about what you have seen in this questionnaire?

This is an open question with a text box.

Figure G.11 shows how this is presented to the subject.

Projectional Drools Survey: Version D

Introduction

Thank you for taking part in this research.

According to Survey Monkey, this survey should take 6 minutes to complete, when we tested it, the average was closer to 10 minutes.

This survey is for the validation section of a research master's project by Paul Spencer at the University of Amsterdam.

The purpose is to determine whether projectional editing can be used to aid the comprehensibility of business rules.

We are using Drools as our example business rules language.

You were selected as you asked or answered a Drools question on StackOverflow, listed Drools as a skill on your LinkedIn profile, or were referred to this survey by someone who previously answered this survey. (please feel free to forward this survey to anyone you know with Drools experience).

It is therefore assumed you are aware of what Drools is.

Projectional editing is a form of writing computer programs directly rather than writing text and having that parsed to create the program. This allows the developer multiple views and editors for the same code.

In this survey, we will present you with a few of these views.

On the following page, there is an animated GIF that will give a small demonstration of what this means.

OK

FIGURE G.1: Screen 1 - introduction text

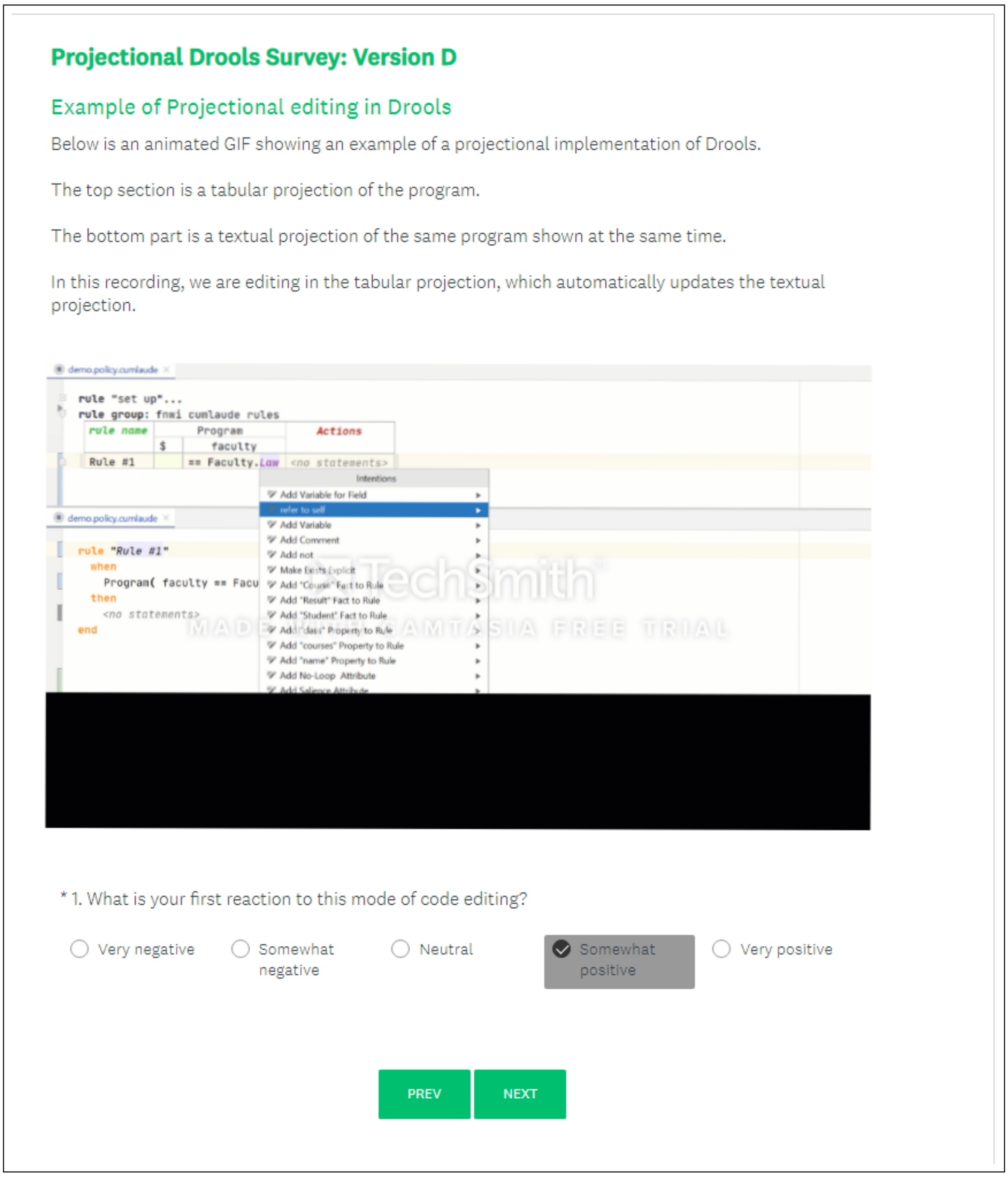


FIGURE G.2: Screen 2 - first impression

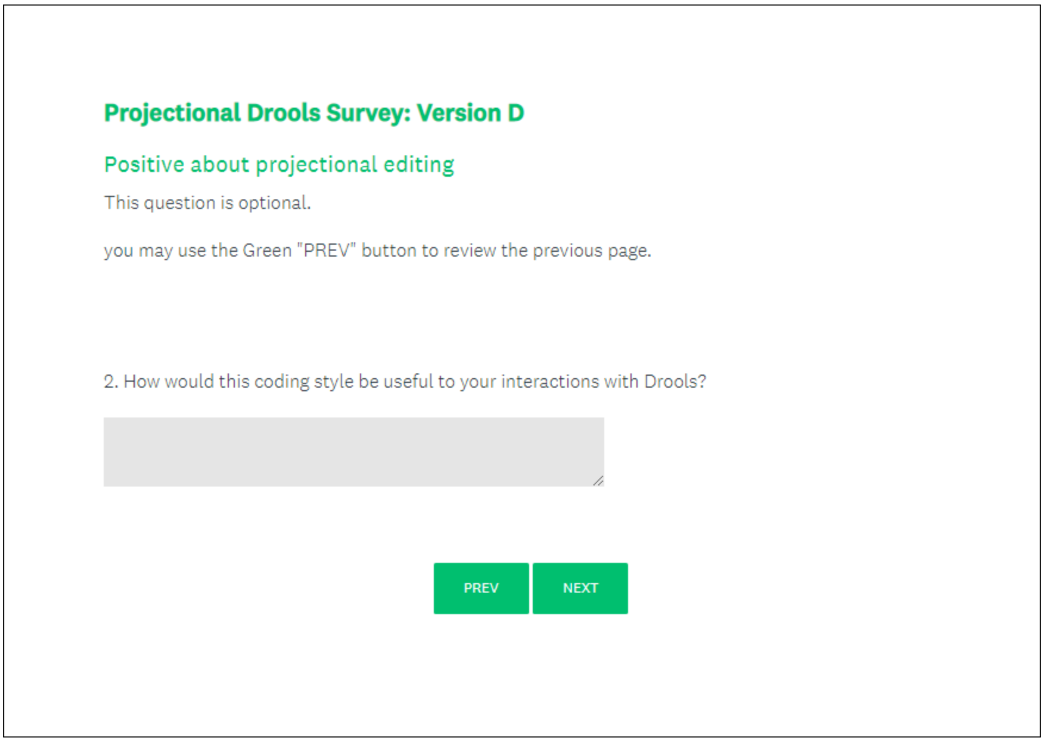
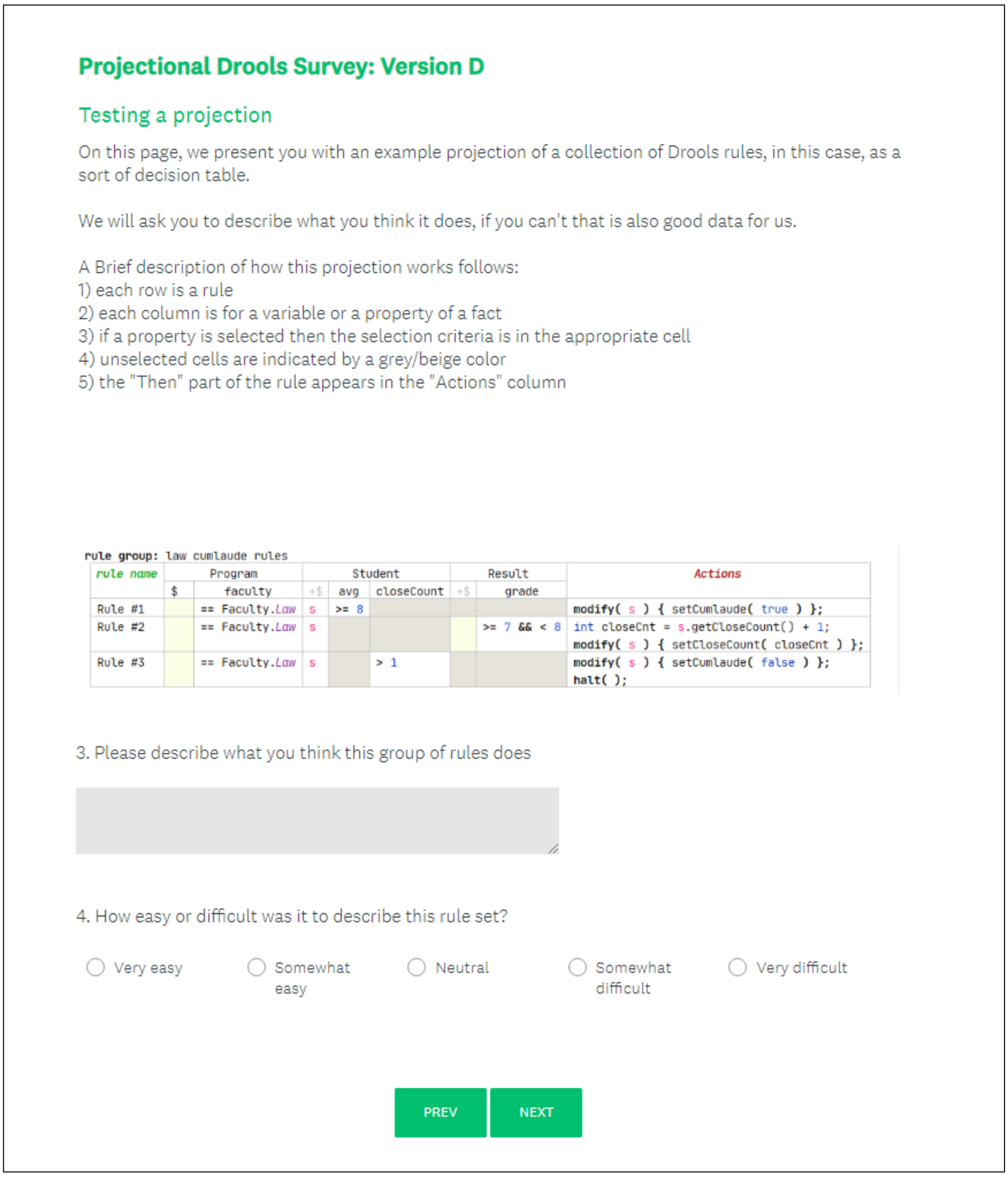


FIGURE G.3: Screen 3 - positive response



Projectional Drools Survey: Version D

Testing a projection

On this page, we present you with an example projection of a collection of Drools rules, in this case, as a sort of decision table.

We will ask you to describe what you think it does, if you can't that is also good data for us.

A Brief description of how this projection works follows:
1) each row is a rule
2) each column is for a variable or a property of a fact
3) if a property is selected then the selection criteria is in the appropriate cell
4) unselected cells are indicated by a grey/beige color
5) the "Then" part of the rule appears in the "Actions" column

rule group: law cumlaude rules

rule name	Program		Student			Result		*Actions*
	$	faculty	+$	avg	closeCount	+$	grade	
Rule #1		== Faculty.*Law*	s	>= 8				**modify**(s) { setCumlaude(true) };
Rule #2		== Faculty.*Law*	s				>= 7 && < 8	int closeCnt = s.getCloseCount() + 1; **modify**(s) { setCloseCount(closeCnt) };
Rule #3		== Faculty.*Law*	s		> 1			**modify**(s) { setCumlaude(false) }; **halt**( );

3. Please describe what you think this group of rules does

4. How easy or difficult was it to describe this rule set?

○ Very easy ○ Somewhat easy ○ Neutral ○ Somewhat difficult ○ Very difficult

PREV NEXT

FIGURE G.4: Screen 4 - describe projection

Projectional Drools Survey: Version D

Testing textual projection

Here we present you a textual projection of Drools rules.

[Note: These are not the same rules as on the previous page]

```
rule "Rule #1"
  when
    Program( faculty == Faculty.FNWI )
    S : Student(  )
    Result( grade < 8, exempted == false )
  then
    modify( S ) { setCumlaude( false ) };
    halt( );
end

rule "Rule #2"
  when
    Program( faculty == Faculty.FNWI )
    S : Student(  )
    c : Course( name == "Thesis" )
    Result( course == c, grade >= 8 )
  then
    modify( S ) { setCumlaude( false ) };
    halt( );
end

rule "rule #3"
  when
    Program( faculty == Faculty.FNWI )
    S : Student( avg >= 8 )
    Result(  )
  then
    modify( S ) { setCumlaude( true ) };
end
```

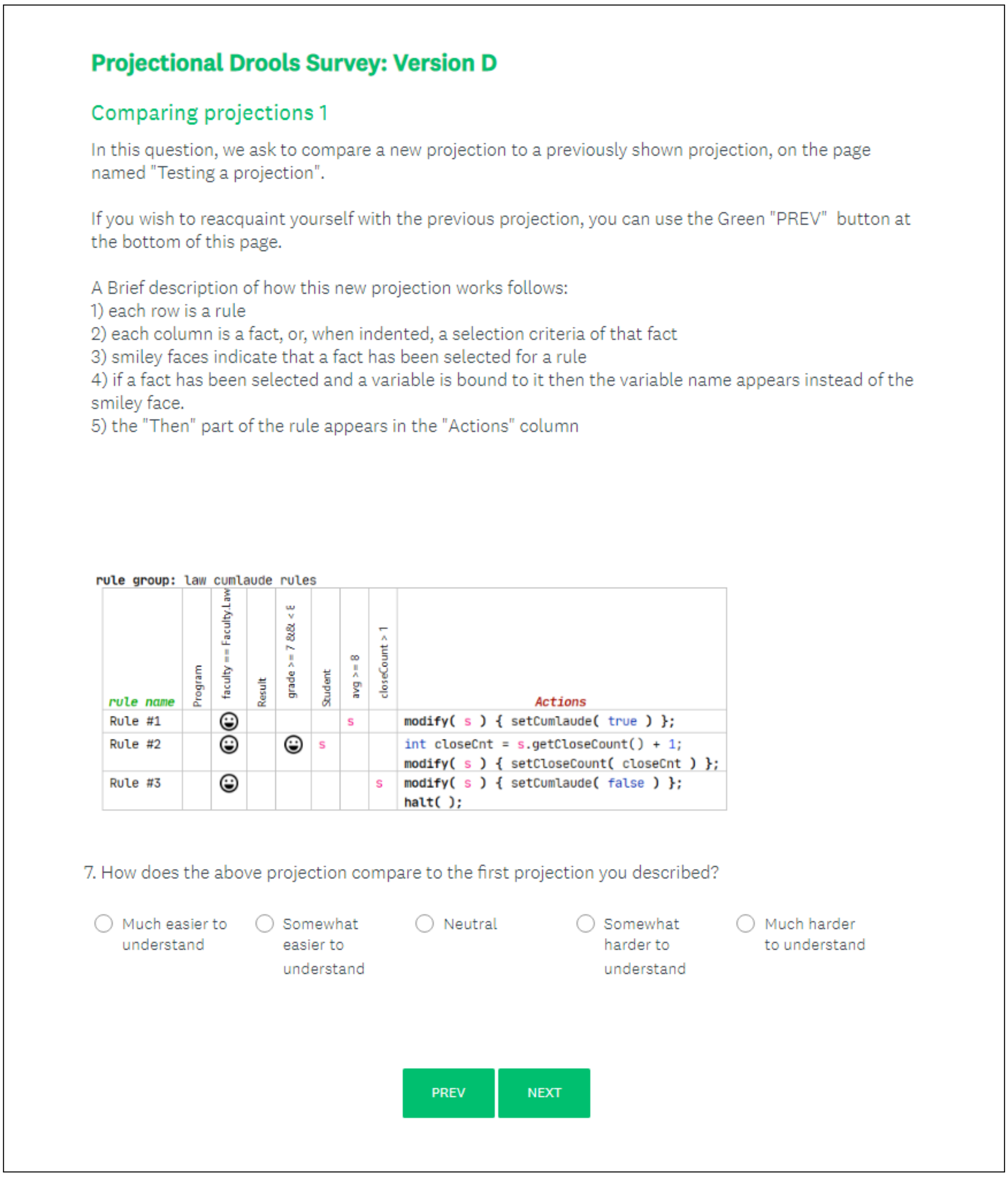
5. Please describe what you think this group of rules does

6. How easy or difficult was it to describe this rule set?

Very easy | Somewhat easy | Neutral | Somewhat difficult | Very difficult

PREV NEXT

FIGURE G.5: Screen 5 - describe text



Projectional Drools Survey: Version D

Comparing projections 1

In this question, we ask to compare a new projection to a previously shown projection, on the page named "Testing a projection".

If you wish to reacquaint yourself with the previous projection, you can use the Green "PREV" button at the bottom of this page.

A Brief description of how this new projection works follows:
1) each row is a rule
2) each column is a fact, or, when indented, a selection criteria of that fact
3) smiley faces indicate that a fact has been selected for a rule
4) if a fact has been selected and a variable is bound to it then the variable name appears instead of the smiley face.
5) the "Then" part of the rule appears in the "Actions" column

rule group: law cumlaude rules

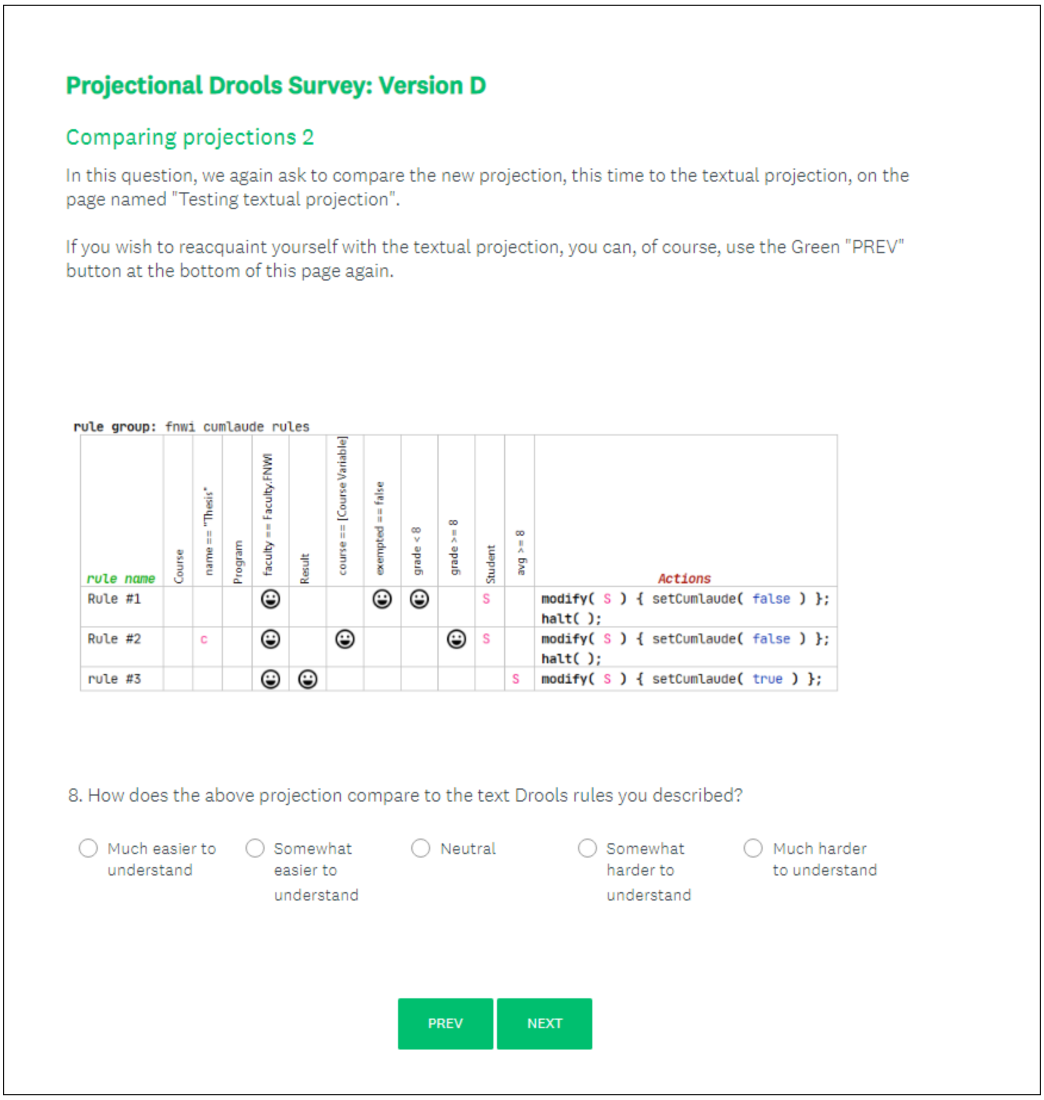
rule name	Program	faculty == Faculty.Law	Result	grade >= 7 && < 8	Student	avg >= 8	closeCount > 1	Actions
Rule #1		😀				s		modify(s) { setCumlaude(true) };
Rule #2		😀		😀	s			int closeCnt = s.getCloseCount() + 1; modify(s) { setCloseCount(closeCnt) };
Rule #3		😀					s	modify(s) { setCumlaude(false) }; halt( );

7. How does the above projection compare to the first projection you described?

○ Much easier to understand ○ Somewhat easier to understand ○ Neutral ○ Somewhat harder to understand ○ Much harder to understand

PREV NEXT

FIGURE G.6: Screen 6 - compare projections



Projectional Drools Survey: Version D

Comparing projections 2

In this question, we again ask to compare the new projection, this time to the textual projection, on the page named "Testing textual projection".

If you wish to reacquaint yourself with the textual projection, you can, of course, use the Green "PREV" button at the bottom of this page again.

rule group: fnwi cumlaude rules

rule name	Course	name == "Thesis"	Program	faculty == Faculty.FNWI	Result	course == [Course Variable]	exempted == false	grade < 8	grade >= 8	Student	avg >= 8	Actions
Rule #1				😀			😀	😀		S		modify(S) { setCumlaude(false) }; halt( );
Rule #2		c		😀		😀			😀	S		modify(S) { setCumlaude(false) }; halt( );
rule #3				😀	😀						S	modify(S) { setCumlaude(true) };

8. How does the above projection compare to the text Drools rules you described?

- Much easier to understand
- Somewhat easier to understand
- Neutral
- Somewhat harder to understand
- Much harder to understand

PREV NEXT

FIGURE G.7: Screen 7 - compare projection to text

Projectional Drools Survey: Version D

Single rule helper 1 - Truth table

Below we present another projection.
This is a truth table projection.
It highlights the conditions that have to be true for a rule to be selected.

The GIF shows the rule selected and the developer pressing the up and down arrow keys to step through the different true (highlighted in green) and false (highlighted in red) fact selections that result in a true outcome.

9. Would this help you with understanding your Drools rules?

- ○ It would really help understanding
- ○ it would somewhat help understanding
- ○ Neutral
- ○ It would add a little confusion
- ○ It would add a lot of confusion

PREV NEXT

FIGURE G.8: Screen 8 - truth table

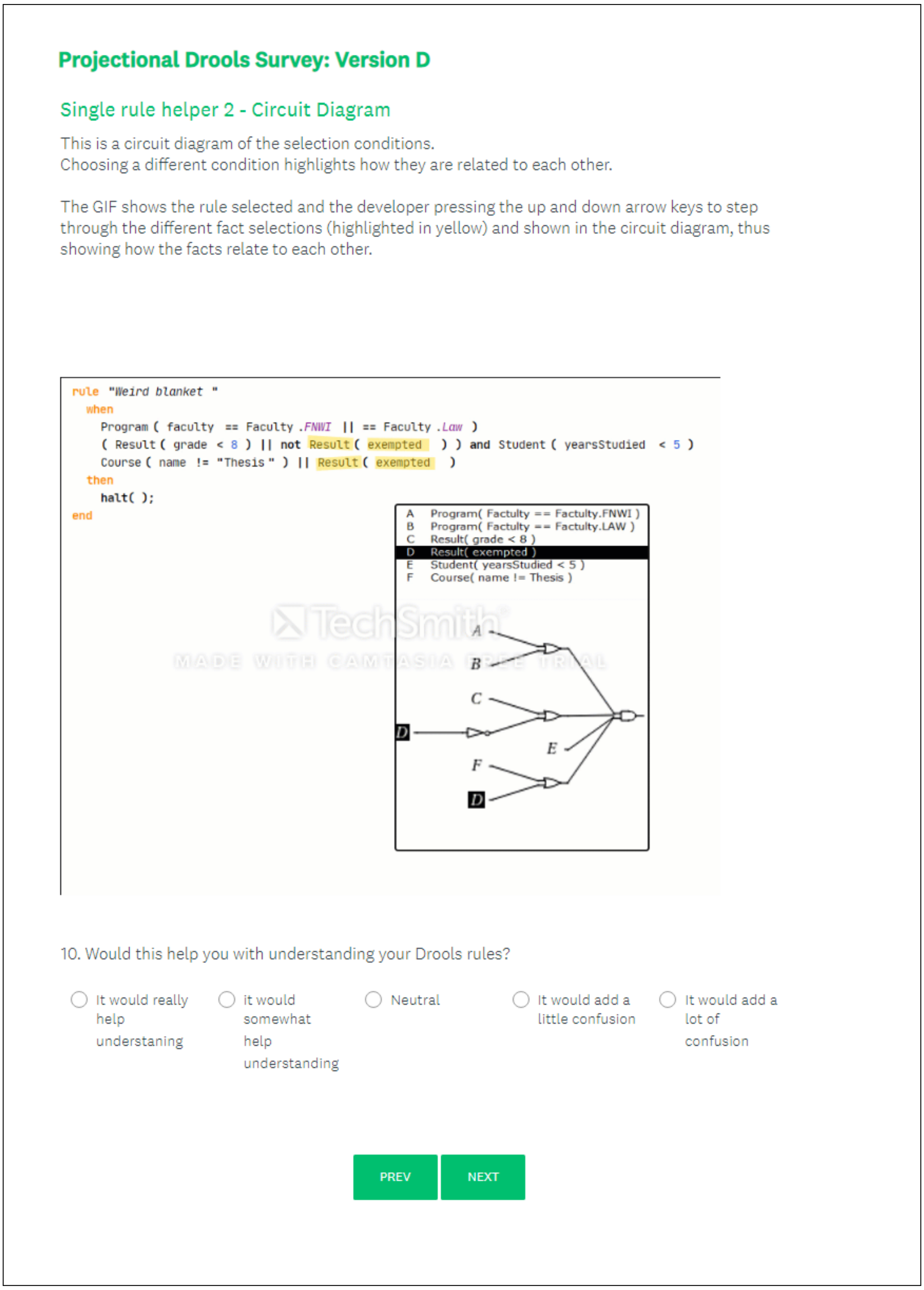
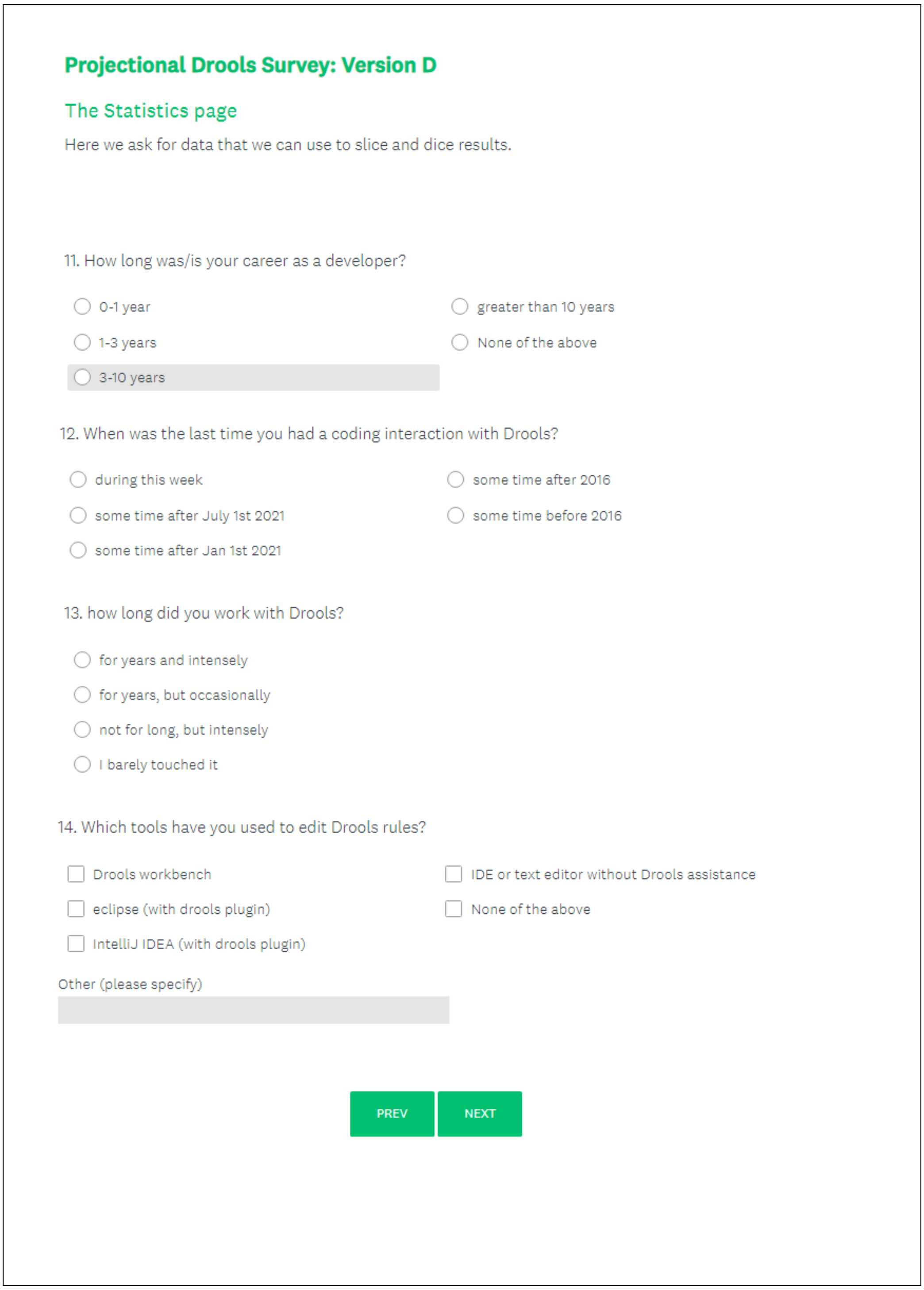


FIGURE G.9: Screen 9 - circuit diagram



Projectional Drools Survey: Version D

The Statistics page

Here we ask for data that we can use to slice and dice results.

11. How long was/is your career as a developer?

- ○ 0-1 year
- ○ 1-3 years
- ○ 3-10 years
- ○ greater than 10 years
- ○ None of the above

12. When was the last time you had a coding interaction with Drools?

- ○ during this week
- ○ some time after July 1st 2021
- ○ some time after Jan 1st 2021
- ○ some time after 2016
- ○ some time before 2016

13. how long did you work with Drools?

- ○ for years and intensely
- ○ for years, but occasionally
- ○ not for long, but intensely
- ○ I barely touched it

14. Which tools have you used to edit Drools rules?

- ☐ Drools workbench
- ☐ eclipse (with drools plugin)
- ☐ IntelliJ IDEA (with drools plugin)
- ☐ IDE or text editor without Drools assistance
- ☐ None of the above

Other (please specify)

PREV NEXT

FIGURE G.10: Screen 10 - personal details page

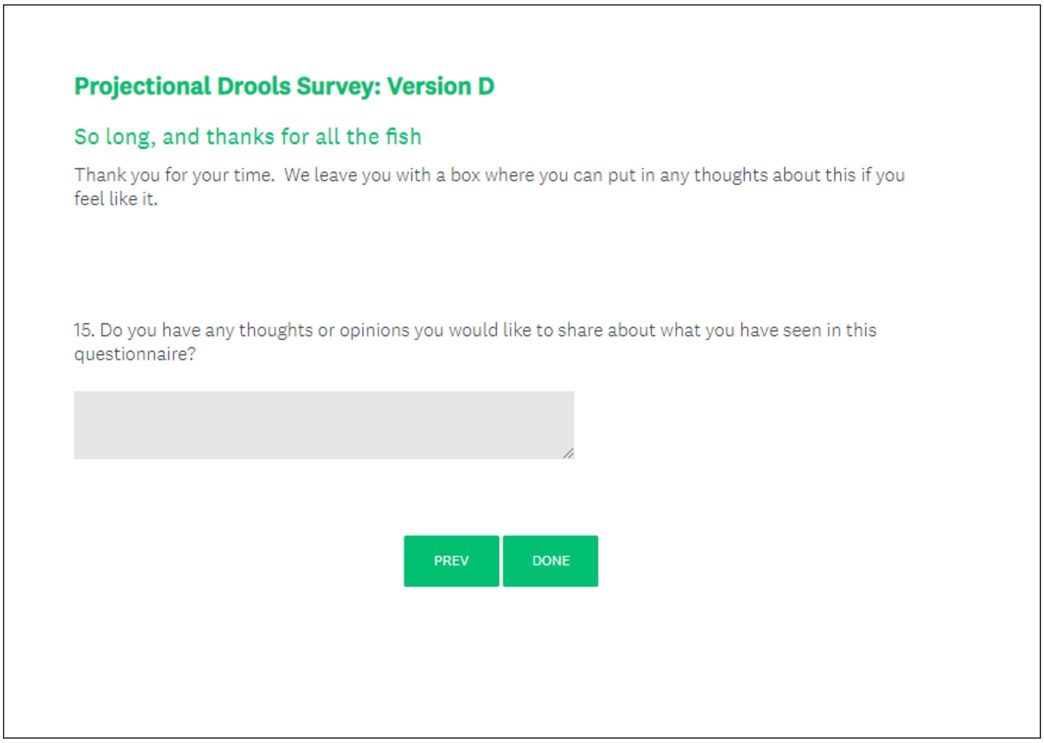


FIGURE G.11: Screen 11 - further comments