eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Degree in Computer Engineering

Computation

Final Degree Project

Solution for the management of several Clinical Practice Guidelines in a domain-independent decision support system

Author

Ainhoa Lizaso Eguileta

2021

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Degree in Computer Engineering

Computation

Final Degree Project

Solution for the management of several Clinical Practice Guidelines in a domain-independent decision support system

Author

Ainhoa Lizaso Eguileta

Instructors

Naiara Muro Amuchastegui

Jordi Torres Piñol

Itziar Irigoyen Garbizu

Ana Jesus Arruarte Lasa

Abstract

Malnutrition is a very frequent and serious problem in humans, even more in the elderly. Advanced age brings with it a series of physiological (e.g., swallowing or chewing problems) and psychological changes that can be considered risk factors for malnutrition. It is triggered by loss, dependency, loneliness, and chronic illness, and potentially impacts on higher morbidity, mortality and the worsen of the quality of life. Without intervention, it presents as a downward trajectory leading to poor health and decreased quality of life. That is why it is essential to assess whether a risk situation exists and to evaluate to what extent it can be evitable.

Therefore, the main objective of this work is to provide nutritional recommendations through a decision support system considering not only the different nutritional needs, also the whole environment of an elderly patient, such as socio-demographic and economic factors (sex, marital status, education...), psychosocial factors (social relationships, family, physical exercise...), and morbidity factors (diseases). Having this in mind, the aim of this work is to provide the most personalized nutritional recommendations.

For this purpose, Clinical Practice Guidelines have been formalized along with experienced nutritionists on the domain within the NUTRIGEP project. The project consists of a product that, in addition to predicting the risk of malnutrition, can prevent it in the geriatric environment, contributing to the good nutritional management of the elderly in order to improve their health condition. Physically it consists of a back-end and a front-end for clinicians and nutritionists, which conceives an integrated solution to support the healthcare professional on the one hand, and to guide the nutritionist in developing personalized diets and preventing malnutrition on the other hand. This work has been focused on the back-end part of this application for the creation, evaluation and management of different rules, using the Drools rule engine and leaning on decision flows, which helped us generating personalized recommendations for patients affected by either one or more pathologies at the same time.

Contents

List Of Figures

List Of Tables

CHAPTER 1

Introduction

1.1 Motivation

Increased life expectancy and the progressive aging of the population lead to an increase in the prevalence of chronic diseases. The sensory alterations, functional disability and social isolation that accompany aging predispose people to inappropriate eating habits and/or imbalances between nutrient intake and the needs of the individual for an optimal physical status. This leads to a number of consequences such as weight loss, alterations of the immune system, aggravation of the underlying disease, longer hospital stays and readmissions, and a poorer quality of life. In addition, the presence of malnutrition in patients at the time of patients on admission to hospital can have a major impact on the evolution and prognosis of their disease and be a cause of increased hospital stay and health care costs.

Therefore, it is necessary to consider new multidisciplinary approaches and the use of nutritional formulas to solve disease-related malnutrition, as it is a health problem of high prevalence and high costs for public health.

To achieve this, the company VICOMTECH, in the eHealth and Medical Devices department, is involved in the NUTRIGEP project with the main objective of predicting and preventing possible cases of malnutrition in elderly patients in addition to customize diets for proper and preventive food management. This work has been developed in this project.

1.2 Objectives of the Project

The general objective of the work is to implement a computer-based solution that offers nutritional recommendations taking into account the comorbid profile of an elderly patient, to provide personalized recommendations that have into account all the different pathologies he/she suffers of.

In order to achieve this general objective, some more specific objectives have been proposed:

1. To study in depth the state-of-the-art of Clinical Practice Guidelines (CPG), Computer-Interpretable Guidelines (CIG), and the Drools rule engine.
2. To learn the basic use of Protégé and how to build a domain ontology using different standardized terminologies (e.g., LOINC, SNOMED CT) to structure knowledge in a flexible, actionable, digital and semantically standardized way.
3. To formalize paper-based CPGs into CIGs to cope with the limitations identified in the state-of-the-art.
4. To learn how to use the Drools and jBPM plugins in Eclipse to be able to work with decision flows to manage multiple rule sets.
5. To learn how to use the Gitlab working environment for the technical project management process.

1.3 Structure of the Document

The work is structured in six chapters as follows:

- Introduction (Chapter 1): The present chapter, which provides an introduction to the work that explains the motivations and lists the different main objectives.
- Project Management (Chapter 2): The management that has been carried out to perform the present work, taking into account the risks that may be encountered, in order to meet the objectives in an orderly way.

- State of the art (Chapter 3): The state of the art related to the problem to be solved is studied in depth in terms of CPG (Section 3.1), CIG (Section 3.2) and the rule engines (Section 3.3).
- Methodology (Chapter 4): The followed methodology to reach the objectives and the output results during this work are described. First, a domain ontology to organize the knowledge in a structurally flexible way and semantically standardized is build (Section 4.1), second several CPGs are formalized into CIGs (Section 4.2) and finally, a solution for the management of these guidelines is implemented in order to give consistent nutritional recommendations using the Drools rule engine using decision flows (Section 4.3).
- Use case (Chapter 5): The NUTRIGEP project, the work done on this use case and the results received are introduced.
- Conclusions (Chapter 6): The personal and academic conclusions drawn from the work are explained. In addition, several pending tasks are proposed to improve the work in the future.

CHAPTER 2

Project Management

This chapter describes in detail the planning process followed throughout the development of the work. This planning process is based on an initial draft schedule made at the beginning, which has evolved as the work has progressed. This has been a necessary step to identify the risks involved in the project and the possible phases of development, and to manage the most valuable resource in a work, which is the time.

To carry out the planning, the work has been divided into three different phases: project management, project development and project documentation. The modules of each phase contain different points of attention that need to be addressed for the most appropriate transition and evolution towards the objectives.

2.1 Description of the phases

In this section a description of each phase mentioned above is made, to clearly see the scheme that has been followed when carrying out the work.

2.1.1 Project Management

The main objective of this phase is to estimate the cost of the different tasks and how the project might evolve throughout its duration. Therefore, it is a phase that is worked at the beginning, estimating the work time during the work and classifying the risks that may be

encountered in order to try to avoid them and, in case they occur, to solve them quickly. In addition, in this phase, apart from estimating the hours, it has been very important to follow up on how the work has been always going and to make sure that the estimate has been met:

- **Planning**: In this task the main points of the work have been decided by estimating its possible duration and the resources needed for its realization, as well as looking for when these objectives can be met and the order in which they are completed. The result has been a set of activities ordered and placed over time with their respective milestones and with a risk management plan to be prepared for them with prior knowledge on how to avoid them. Finally, the scope of the project has been determined.

- **Monitoring**: The objective of this task has been to constantly check that the objectives assigned at the beginning are being completed well and on time. In this section new risks could be found and even new objectives could be created.

- **Communication**: This part unites the collaboration of the student and the tutor to see if the plan is being followed. In this part, changes in the work have been communicated to the mentor in order to reach an agreement of both. These issues have been discussed in meetings and at the end the tasks have been decided until the next meeting, which have been usually short term intentions.

2.1.2 Project Development

The work has been strongly oriented towards a decision support system based on Artificial Intelligence (AI) methods to provide the user with diet and/or supplement recommendations. Most of the development phase has been focused on researching the current state-of-the-art regarding how decision support systems work, the main difficulties caused by clinical practice guidelines both in paper and digital formats and the possibilities to solve them.

Once all this have been studied, the practical application has been developed which includes all the steps learned in the theoretical part to see in action the solution created and see the results obtained.

2.1.3 Project Documentation

The last part consists of the work report, where the most relevant knowledge and information about the work done is collected, both for the theoretical part and for the practical application. The amount of research in this phase has been quite long to ensure that every aspect of the methodology can be fully understood and provide an enriching and valuable insight.

2.2 Estimations

After defining the three phases that make up the work, the initial time estimate and the amount of time ultimately required to complete the tasks have been assigned. Table 2.1 shows the estimates made both for the phases in general and for the tasks within each phase.

	Estimated time (h)	**Final time (h)**
Project Management	**50**	**45**
Planning	20	15
Monitoring	20	15
Communication	10	15
Project Development	**180**	**200**
Knowledge Acquisition	100	100
Implementation	80	100
Project Documentation	**70**	**85**
Report	70	85
Total	**300**	**330**

2.1 Table: Dedication table

2.3 Risk analysis

The risk management plan has been a progressive plan to prevent risks, including new ones, swapping other ones or discarding some others. Therefore, the risk management plan has evolved to adapt to each moment. Particularly, the risks have been identified in the monitoring process and later added or modified after a meeting with the directors to discuss the potential consequences and how to prevent them.

One risk has been the loss of information due to problems with the working machine. To avoid this, it have been used cloud services (Drop Box for files, Gitlab for code, and Overleaf for the documentation of the work) to store the information.

In addition, problems may arise when adding new functionalities or use different libraries. To avoid wasting time in restructuring the code to return to the previous state, a version control has been performed, where a current version of the application has been saved as improvements were added, to ensure that if the code failed, the previous version has been accessible.

Finally, another major risk of this project has been the consequences of the COVID pandemic. Due to the fact that the project was carried out in a company, students were sent to work from home to deal with the pandemic and this could cause communication, organization and learning problems, since working at home is always more confusing. In order to avoid any possible misunderstandings, it has been decided to organize meetings every week to work progressively and to always have a goal to achieve before the end of the week.

CHAPTER 3

State of the Art

This section will give an overview of the state of the art of clinical practice guidelines (CPGs). The importance of formalizing CPGs into Computer-Interpretable Guidelines (CIG) and how to manage one or more guidelines in a Clinical Decision Support System (CDSS) using a rule engine will be analyzed.

3.1 Clinical Practice Guidelines

CPGs are defined as a set of systematically developed recommendations aimed at standardizing the clinical care and optimizing patient's outcomes [1]. They are based on a systematic review of the latest clinical evidence and an assessment of the benefits and harms of alternative care options. CPGs are developed by multidisciplinary teams, who review the evidence comprehensively and systematically, evaluate the quality of the information, and present specific recommendations. Recognition of the need for CPGs has been caused by the following several observations: growing evidence of substantial unexplained and inappropriate variations in clinical practice patterns, concern that resource limitations could reduce the possibility of providing high quality health care, and the difficulty clinicians have in assimilating rapidly evolving scientific evidence into their practices [2].

The aim of clinical guidelines is to guide professionals and clinicians involved in the decision-making process improving the quality of care, limiting unjustified variations in clinical practice, and reducing healthcare costs [3].

However, paper format CPGs do have some problems by their own. On the one hand, many guidelines discuss about the same topic and this makes them inconsistent and with discrepancies when giving advice on the same pathology [4]. On the other hand, paper-based guidelines have not been used on a large scale by doctors because of several barriers. The most relevant are the accessibility of the knowledge of the guideline when it is long and complex and a quick and instantaneous response is required, the difficulty of analyzing and observing several therapeutic guidelines at the same time, the need to constantly update the guideline information, and the importance of having current patient information at all times [5].

Seeing these inconveniences, the development of CDSSs was strengthened. To this purpose, it was necessary to formalize the CPGs as CIGs in order to be able to apply them as a knowledge base of CDSSs. These CDSSs combine the formalized knowledge of guidelines with up-to-date clinical data from patients to provide them with specific advice at the point of care, which increases the potential to influence physician behavior compared to the exclusive use of narrative guidelines [3]. This makes clear the benefit of improving medical and professional knowledge by providing easy access to scientific resources, presenting reminders, and providing useful information for desirable decision making with minimal errors [6].

3.2 Computer-Interpretable Guidelines

CIGs are increasingly being applied in diverse areas, as knowledge base of CDSSs, covering a wide range of clinical settings and tasks. These digital documents promote the acceptance and implementation of guidelines in daily practice, as CDSSs are capable to monitor the actions and observations of care providers and provide guideline-based advice at the point of care [7].

In this section, the creation of CIGs has been analyzed. In subsection 3.2.1 some of the general guideline models have been mentioned, distinguishing the model and the language in which the guidelines are specified and giving a brief description of each. In subsection 3.2.2 the obstacles that may be encountered when creating CIGs have been discussed.

3.2.1 General Models

When formalizing a CIG, the following two parts should be considered: the model and the language in which the guidelines are going to be specified.

The model is the main characteristic of any guideline approach. It must be able to represent several types of guidelines that differ considerably in complexity and level of abstraction. The model must contain a set of building blocks (primitives) used to construct guidelines, such as tasks, rules, nodes and framework. The guideline model must be supported by a formal language that specifies the model primitives mentioned above. Typically, a guideline language consists of two parts, a control flow language and an expression language. The control flow language specifies the structure of the guidelines in terms of the model primitives and their (temporal) relationships, while the expression language usually describes the decision criteria [8].

Peleg et al. [9] compared five CIG models - Asbru, EON, GLIF, Guide and PROforma - based on eight different dimensions which are the CIG modeling languages, knowledge acquisition and specification methodologies, CIG integration with organizational workflow, validation and verification, execution engines and supportive tools, exception handlings, CIG maintenance and finally CIG sharing. Each of the CIG frameworks studied performs the following aspects: (1) represent a guideline as a set of tasks and relationships between them, (2) provide expression/criterion languages to specify decision criteria such as comparison operators, (3) specify the guideline's intention or goal, (4) represents medical concepts and (5) represents patient information and maps to an electronic medical record.

Asbru is a CIG model that is used to express guidelines as time-oriented skeletal plans that can be instantiated for each patient. This model allows the designer to represent the prescribed actions of a plan, as well as process intentions and outcome intentions. Time-oriented actions, conditions and intentions are expressed as patterns to be maintained, achieved or avoided, during or at the end of a plan. A plan consists of a name, a set of arguments, including a time notation (representing the temporal scope of the plan), and five components: preferences, intentions, conditions, effects, and a plan body that describes the actions to be executed. It may also have sub-plans that have the same structure.

EON is a model and execution system that uses a task-based approach to define decision support services that can be implemented using alternative techniques. It is also the first energy-aware programming language. EON allows programmers to build programs

from code written in a variety of languages, including nesC and C. It provides a simple way to associate particular control flows with abstract energy states that represent the energy available in the system. Its own execution system only executes flows that the programmer has marked as appropriate for the given energy state. Thus, the programmer can easily write programs that provide different functionality or data quality depending on the current and future energy availability.

The Guideline Interchange Format (GLIF) is a computer-based format to distribute guidelines across different institutions and systems. It represents a clinical guideline as a network of steps resembling a flowchart where steps can be implemented branching logic or execute actions [10]. GLIF consists of classes, their attributes and the relationships between classes, all of which are necessary to model clinical guidelines [11]. These guidelines are represented as flow diagrams of nodes represented by an abstract class called `Guideline_Step`. This class has the following subclasses: `Decision_Step`, `Action_Step`, `Branch_Step`, `Synchronization_Step` and `Patient_State_Step`. Its expression language was originally based on the Arden Syntax and its default medical data model is based on the HL7 Reference Information Model (RIM). But a subsequent object-oriented language, GELLO, is being refined for consideration as an HL7 standard. GELLO grows as a standard query and expression language for decision support. It provides specialist physicians with the ability to customize their existing systems and create flexible, purpose-built decision support systems. In addition, it uses an abstract "virtual medical record"(vMR) so that the same GELLO code can run on multiple systems accessing data stored in different formats [9].

The GUIDE model allows the developer to formally represent the guides and protocols. The representation approach is flow-based. The main components are tasks (rectangles) and decision points (diamonds). Given the complexity of care delivery processes, it adopts a hierarchical representation. Thus, some tasks can be expanded to a lower, more detailed level. This model produces four different XML data structures separating the general properties of the CPG, the set of medical terms (such as clinical observations, diagnoses and therapies), the set of abstractions to represent complex concepts and the GPC flow indicating the activities and the decision processes.

PROforma combines logic programming and object-oriented modeling and is formally grounded in the RL language. One aim of the model project is to explore the expressiveness of a deliberately minimal set of modeling constructs. It supports four tasks: actions, compound plans, decisions, and enquiries. All tasks share attributes describing goals, control flow, preconditions, and postconditions.

A summary of the analyzed languages is captured in the following tables. On the one hand, Table 3.1 presents the representation of each language and on the other hand, Table 3.2 shows the structure and patient data modeling of each language. It can be observed that in some boxes n/a is shown, this means that this language information is not available in the publications.

Guideline Model	**Representation Primitives**			
	Decision	**Action**	**Patient State**	**Execution State**
Asbru	Condition, preference	Plan	Temporal patterns	Plan state
EON	Decision step	Action, activity	Scenario, activity state	No
GLIF	Decision step	Action step	Patient state step	No
GUIDE	Decision	Task, wait, monitor	(implicit)	n/a
PROforma	Decision	Action, enquiry	n/a	Task state

3.1 Table: Representation of languages

Guideline Model	**Structure for Primitives**		**Patient Data Modeling**
	Temporal Constraints	**Nesting**	
Asbru	Plan-body	Plan	n/a
EON	Flowchart	Subguideline	EMR ontology
GLIF	Flowchart	Subguideline	Three-layer domain ontology
GUIDE	Flowchart	Task	Relational
PROforma	Constraints satisfaction graph	Plan	n/a

3.2 Table: Structure of languages

In conclusion, seeing that each model can provide different characteristics and different functionalities, it can be said that there is no certain standardization language for the representation of CIGs. Nevertheless, is believed that it would be highly recommended to apply Semantic Web Technologies (SWT) to process data in a more effective and efficient way, to create a suitable framework for interoperability between systems and also to integrate data from various sources.

3.2.2 Barriers for the creation of CIGs

Previously the most common models to represent CIGs has been studied. Knowing that this work is based on a rule-based model, several disadvantages of its implementations must be taken into account.

On the one hand, semantic problems occur not making distinctions between the different types of knowledge presented. This is that the messaging standards used in healthcare use different terms for the same concept, often resulting in clinical misinterpretation, poor knowledge management and misdiagnosis of the patient's disease. To solve this, computational ontologies are used. An ontology consists of a set of concepts organized by their relationships. The concepts and relationships included must describe the agreed knowledge of a domain to be followed by both humans and machines [12]. Ontologies are designed using standardized codifications that guarantee the interoperability of the implemented knowledge and its univocal interpretation, since it allows the representation of concepts with stable and single codes. Some of the known codifications for the representation of the health domain applications are SNOMED CT [13] and NCI thesaurus [14]. They have a great expressiveness in specifying the concepts, properties, constraints, and type of relations included. They could be reused for the design of ontologies, due to the fact that their declarative formalism based on logical descriptions enables them to discover new information by inference.

On the other hand, syntactic problems are involved with guidelines that are developed over long periods of time not being compatible, such as those defined for chronic patients. In order to overcome these limitations, different standards have been created called messaging standards, used to encode and exchange health information. Messaging interoperability standards serve to ensure that the transmission of information between systems is consistent. The main function of these standards is to define the structure, data type and format so that systems, using this clinical information, can exchange data securely and efficiently. An example of these standards is HL7 [15].

3.3 Rule Engines

In the previous section the creation of CIGs has been analyzed in order to apply them as knowledge base of CDSSs. In this section the functionality of these systems and a brief comparison of different rule engines is presented.

Most CDSSs consist of three parts according to Miller et all. [16]. These parts are the knowledge base, the inference or reasoning engine, and a mechanism to communicate with the user. The first part, as explained in section 3.2, would be the knowledge base which adopt different formats such as a set of clinical rules, models obtained from data, etc. The second part of the CDSS is called the inference engine or reasoning mechanism, which contains the formulas for combining the knowledge base with data entered to analyze. Finally, there must be a communication mechanism, a way of getting the data into the system and getting the output of the system to the user who will make the actual decision. In this work, clinical rules based on domain-specific clinical guidelines are used as knowledge base, and patients' data is entered into the system to generate the recommendations.

Focusing now on the inference engine of a CDSS, it performs functions such as interpreting and executing the guidelines encoded in the specific representation formats. Since the core logic of clinical guidelines consist of complicated rule sets, commercially available rule engines have been adopted on a wide-scale basis [17].

Some studies have suggested the adoption of the workflow concept in clinical guidelines [9, 18]. It is very promising because it facilitates the decision support as it can analyze patient specific clinical information using latest available evidence. By implementing a workflow and automating the tasks contained in a guideline, all contents of the guidelines are included within a pre-established order and hierarchy. Thanks to the workflow, processes are automated and organized in a way that is much more accessible to all users. Therefore, in this work we integrate the use of workflow in rule engines.

In order to select the most suitable workflow and rule engine, the following elements are considered according to Lee et al. [17]:

- Integrity: In order to achieve fast response and correctness of execution, the workflow and the rule engine should be easily integrated.
- Extensibility: The framework of engines should be based on a well-known architecture so that its components can be easily added or reconfigured.

- Open source: The main advantage is the possibility of sharing, modifying and studying the source code of a computer system with the collaboration of different users.

There are different open-source rule engines such as CLIPS, Drools, NxBRE, OpenRules, Jena, Hammurapi Rules, OpenL Tablets and CodaServer that implement a rule engine and provide different management mechanisms. However, only Drools, OpenRules and OpenL Tablets have the main components of a Business Rules Management System (BRMS) [19]. A BRMS is a software solution used to define, deploy, execute, monitor and manage rules and decision logic. With a BRMS, rules and decision making are automated across an organization's processes. It can distinguish the relationships between various rules and associate the rules with technology solutions that perform the required functions.

The three main components are (i) a development environment (ii) the rules repository and (iii) a rules execution environment. Rules are stored in a repository instead of embedding the rules as code within the application. In this way, the rules can be accessed by more than one application and are available for reuse. An execution environment is a hardware and software infrastructure that enables the operation of a code base in real time. In a BRMS, this allows applications to invoke the applicable rules and execute them using a rule engine. In addition, a BRMS provides development tools for users to define and manage business rules. In this way, users can develop business rules without writing code. Table 3.3 shows a summary of the technological strengths of the Drools, OpenRules and OpenL Tablets components.

BRMS	Rules engine	Management mechanisms
Drools	Performs inference with forward and backward chaining. Implements Rete algorithm and JSR-94 standard. Handles large number of rules, orders/transactions and users	It allows editing rules in Java, XML, Drools native language (drl), domain specific languages and Excel. It has Eclipse plug-in and web editor. Publishes rules as web services. Generates reports. Allows to create and execute test cases. Facilitates rule debugging. Allows to create organizational vocabulary
OpenRules	Performs forward chaining inference. Implements the Rete algorithm and a specific algorithm for optimization problems. Implements the JSR-33 standard	It can represent rules in XML language and decision tables in Excel, OpenOffice or Google Docs. It has an Eclipse plug-in and Web editor. It has a Web application to edit rules. Generates reports. Allows the creation and execution of test cases. Facilitates the creation of an organizational vocabulary
OpenL Tablets	Performs forward chaining inference. Implements the Rete algorithm	It has Web editor, publishes rules as Web services. Makes it possible to edit rules in simplified Java language and in Excel tables. It has a plug-in for the Eclipse program. Allows creating and executing test cases

3.3 Table: Drools, OpenRules and OpenL Tablets tool properties

As seen on Table 3.3, all platforms have many ways to write rules (tables, an own language, XML etc.), have different rules execution environments and have Open-Source licenses, but only Drools platform has both, an advanced knowledge representation language and an powerful inference engine that can perform the forward and backward inference. In addition, by creating Domain Specific Languages, understanding, and writing rules becomes much easier, since it is written in natural language with the <if-then> expression. It is also worth emphasizing that as mentioned before some studies have suggested the adoption of the workflow concept in clinical guidelines and Drools contains the DroolsFlow module that develops workflow activity types for rule management. Drools contains within it the plugin to use workflows so there is no need to integrate any other language (e.g., CLIPS, Asbru, PROforma, ...) to define the guidelines compared to the other two rule engines. In conclusion, from our point of view Drools is the most suitable engine for this work.

CHAPTER 4

Methodology

This work is based on the work done by N. Muro et al. [20], who developed a decision event structure to collect all the information related to the decision-making process. Its architecture is based on three modules which are (i) the definition of a decisional event in a processable structure, (ii) the CPG formalization module and (iii) the semantic validation module. In this work, this structure has been extended adding a tool to manage (i) more than one rule group in a single CPG, and (ii) the handling of several CPG formalized as DRLs at once when evaluating a comorbid patient. This section will describe the followed methodology and the output result work done during this research project to reach the objective.

In order to manage guidelines, they must first be formalized and, for this, it is necessary to have a model of the domain to avoid semantic interpretation errors. Therefore, before formalizing and creating a solution, an ontology that will be used during the formalization of the rules has been generated.

4.1 Creating the ontology

Ontologies require a logical and formal language to be expressed. Numerous languages have been developed for this purpose such us Ontology Inference Layer (OIL), Web Ontology Language (OWL), Resource Description Framework (RDF) or RDF Schema (RDFS). The presented work interacts with a ontology formalized in RDF language with

a connection to the formalization of CIG through the Protégé tool [21], which implements this language based on frames (taxonomies of classes and attributes), with a significant expressive power. Protégé allows:

- Modeling an ontology of classes describing a particular topic
- The creation of a knowledge acquisition tool to collect knowledge
- Entering specific cases of data and a knowledge base

In order to create the ontology, the following steps have been taken:

Step 1. *Determine the domain and scope of the ontology*. This step helps to determine the extent of the model, defining the scope of the ontology with respect to its specific domain.

Step 2. *List important terms for the ontology*. In this step it is necessary to write a list with all the variables that will potentially serve for the ontology. Variables are extracted from the description of the application domain.

Step 3. *Define classes and class hierarchy*. There are several possible approaches to develop a class hierarchy: a top-down development process, bottom-up development process, and a combination of top-down and bottom-up approaches. In this project, a top-down development process has been used, which consists of starting with the definition of the most general concepts in the domain and subsequent specialization of the concepts.

Step 4. *Define class properties: slots*. Once some of the classes are defined, the internal structure of the concepts must be described

Step 5. *Define the facets of the slots*. Slots can have different facets that describe the type of value, supported values, the number of values (cardinality), and other characteristics of the values that the slots can take.

4.2 Formalizing CPGs

The formalization of CPGs into CIGs has been done starting from the work done by N. Muro et al. [20] which consists of a java-based framework that facilitates this process. CPGs can be written in many document-based format (e.g., .drl, .xml, .json), but in this work they have been translated to JSON documents. This way it is possible to have the

CPG knowledge in a computerized form. This object can be edited at any time in the future to include new knowledge contained in new CPGs versions.

The framework has an object called Condition. This object stores (i) the name of the clinical variables to be evaluated, (ii) the mathematical operator (i.e., >, >=, =, <=), and (iii) the value imposed by the condition to be evaluated.

The formalization process starts with defining the conditions that make up the rule. The variables of the conditions are entered, and the ontology is queried to obtain the information regarding its type and possible values. For example, if the first variable is "Age", the condition value will be restricted to the possible values defined for that variable in the ontology (i.e., natural numbers). Once the condition is fulfilled, a binary operator can be introduced (i.e., AND, OR) for including more conditions or end building the rule by defining the recommendation. Figure 4.1 illustrates the condition format in the JSON document.

4.1 Figure: Condition format in JSON document

Once all the rules have been written in JSON, the object is parsed to generate an instance of a Java Guideline object. This object is parsed into a DRL file using a template that has the structure shown in Figure 4.2. The Java object contains the needed parsing methods to translate the rules into the Drools Rule language. The translated rules are then inserted into the DRL file using the mentioned template. The parts of the rule (name, data object, conditions and recommendations) are entered in the template as a Java Map and its values are mapped in the template, where each attribute will be replaced by its value.

```
dialect "java"

template "RuleTemplate"

rule "@{row.rowNumber} - @{name}"
when
    @{object}(@{conditional} )
then

    recom.add("@{action}");
    debug(drools);

end


end template
```

4.2 Figure: Rule template based on DRL language

4.3 Multiple rules management

So far, it has been introduced how to create the ontology and how rules have been formalized in CIGs. In this section it is described how the work done by N. Muro et al. [20] has been extended in order to add a new tool that allows to manage different CIGs with the help of the decision flows.

The architecture has been created in Java following Drools rule engine, which activates the rules in a production environment at runtime. These rules come from the formalized CIGs and inputted patient data are evaluated to return the treatment that best fits the clinical case.

Drools is composed of different modules that work on different aspects. These modules are Drools expert, Drools fusion, Drools flow, Drools guvnor and Drools solver. Among the modules, the ones that are interesting for this work are the following [22]:

- Drools expert: is basically the rule engine. Stores, processes, and evaluates data to execute the business rules or decision models that are defined. The basic function is to match incoming data, or facts, to the conditions of rules and determine whether and how to execute the rules.

- Drools flow (jBPM): provides workflow capabilities to the Drools platform. A business process or workflow describes the order in which a series of steps need to be executed, using a flow chart. This makes it much easier to describe a complex

composition of various tasks. It allows end users to specify, execute and monitor their business logic.

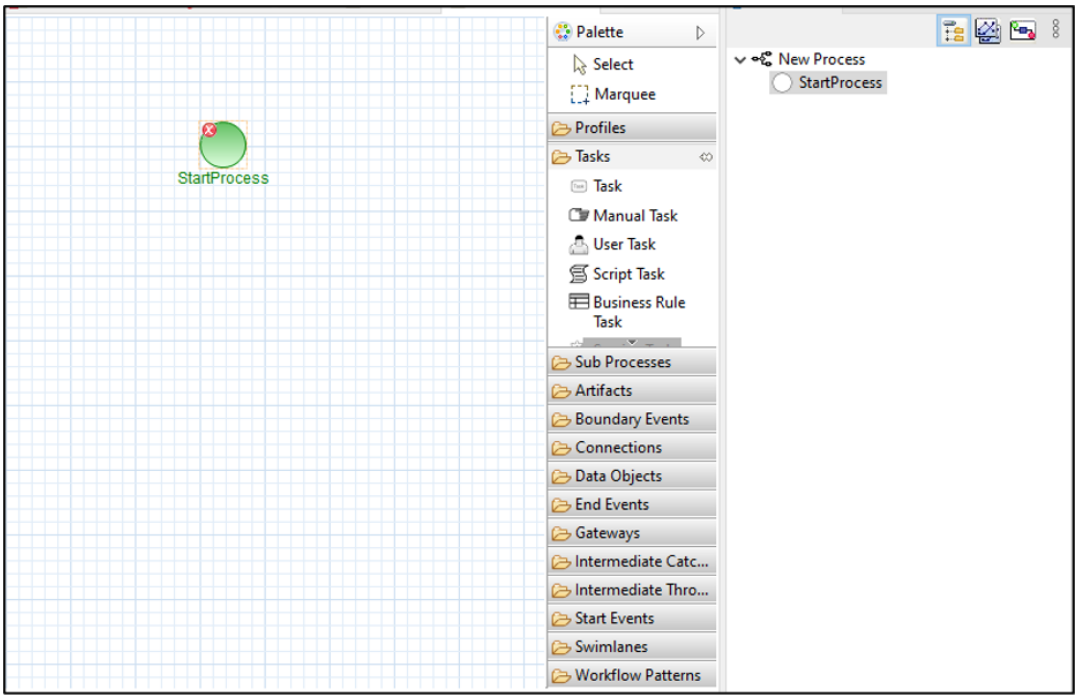
The Drools Flow module has been the focus of this project. The Drools Flow module allows to visually define the flow in which business rules and their associated actions are executed. Drools Flow is another way to have human-readable rules. It allows to define the flow of execution between rules. It can execute arbitrary actions at specific points inside the flow and becomes very interesting because with this a patient can be assessed at each moment with the characteristics and needs that he/she has at that moment, which gives us the possibility to execute certain DRLs according to the patient's data.

There are two things that are necessary to run processes from an application: (1) create a Knowledge Base containing the process definition, and (2) start the process by creating a session to communicate with the process engine and start the process.

4.3.1 Creation of a process

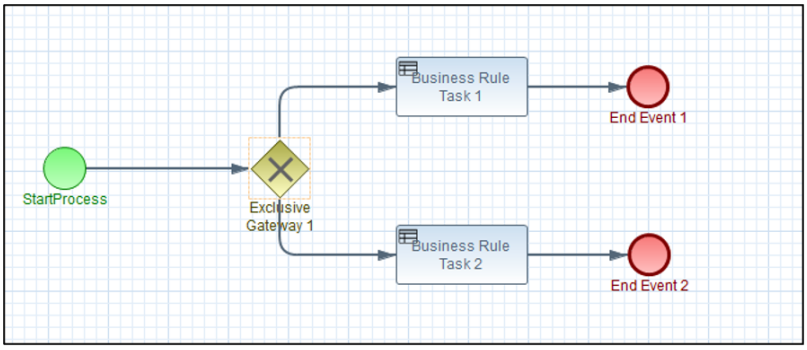
When creating the Knowledge Base, many processes can be created using three different methods: using the graphical RuleFlow editor in the Drools plug-in for Eclipse, as an XML file, or directly creating a process using the Process API. In our case, the first method has been implemented as it is the most visible and easy to understand. Figure 4.3 shows the graphical RuleFlow editor. It allows to create a process by dragging and dropping different nodes on a canvas and editing the properties of these nodes. The process itself has the following properties:

- Id: the identification label of each process
- Name: the display name of the process
- Version: the version number of the process
- Package: the package or namespace where the process is defined



4.3 Figure: New RuleFlow process

A rule flow process is a flow diagram in which different types of nodes are linked by connections (folders column on the right in Figure 4.3), but the ones that have been studied are the ones used in this project, the Start, End, Business Rule Task and Gateway nodes. These modules have been chosen mainly because the Rules Task node makes it possible to separate all the rules into groups that best suits us. Gateways are also used because they are the element needed to guide an execution of patient's data into one group of rules or another. Next, a more detailed description of each of the nodes that have been used and an example of a simple process are presented:



4.4 Figure: A simple process example

Start: The start of the ruleflow (green circle in Figure 4.4). A ruleflow should have exactly one start node, which cannot have incoming connections and should have one outgoing connection. Whenever a RuleFlow process is started, execution will start at this node and automatically continue to the first node linked to this start node, and so on.

End: The end of the ruleflow (red circle in Figure 4.4). A ruleflow should have one or more End nodes. The End node should have one incoming connection and cannot have outgoing connections.

Business Rule Task: Represents a set of rules (rectangles in Figure 4.4). The rules are evaluated when the node is reached. These rules are grouped using the ruleflow-group attribute in the header as can be seen in Figure 4.5. This greatly reduces the execution time by not needing to review all the rules and only analyzing the ones that correspond to the group. When a BusinessRuleTask node is reached in the ruleflow, the engine will start executing rules that are part of the corresponding ruleflow-group (if any). Execution will automatically continue to the next node if there are no more active rules in this ruleflow group.

```
rule "1 - R01"
ruleflow-group "alzheimer"
when
    HashMap((Age >= 65.0) && (Age <= 69.0) && (Sex == "Woman")
    && (DesnutritionRisk == "Medium") && (IntakeValuation == "Appropiate")
    && (Texture == "Normal") )
then
    recom.add("{\"guidelineName\":\"NUTRIGEP Nutrition guideline\",\
    "guidelineID\":\"NUTRIGEP\",\"ruleGroupName\":\"Alzheimer\",\"ruleGroupID\
    ":\"NUTRIGEP_RG_01\",\"recommendations\":[{\"usability\":0.25,\"strengthOfRecomm\
    ":0.74,\"sourceRuleID\":\"NUTRIGEP_RG_01_RL_01\",\"orderSet\":[{\"type\":\"Diet\",\

    "..."

    "stepNumber\":\"0\",\"value\":\"PerWeek\",\"information\":\"\"}]}]}");
end
```

4.5 Figure: Ruleflow-group atribute

Gateway: This node allows to create branches in your rule flow (diamond in Figure 4.4). There are two types of Gateway nodes currently supported:

- INCLUSIVE means that the control flow will continue in all outgoing connections simultaneously. This implies that the execution process will be follow several paths at the same time and multiple drl files will be evaluated simultaneously.
- EXCLUSIVE means that exactly one of the outgoing connections will be chosen.

The decision is made by evaluating the constraints that are linked to each of the outgoing connections. Always at least one of the outgoing connections will evaluate to true at runtime (the ruleflow will throw an exception at runtime if it cannot find at least one outgoing connection). This implies that the execution process will follow only one branch.

The conditions to choose a branch or several branches in both inclusive nodes and exclusive nodes are specified within the node properties by writing them with the same syntax as in a rule contained in the DRL file, or in plain java language.

4.3.2 Creation of a session

One of the first steps when creating a Drools project is to create a session that is in charge of preparing the execution and setting up the ecosystem of rules associated with this execution. Adding a process to the session changes a little bit the way of creating the session itself, since it must be loaded with its identifier (see line 2 in Figure 4.6) to direct the session to the package that stores the knowledge base (process file and DRL files).

The process it is only executed if it is explicitly indicated. This is because more than one process could be defined in the Knowledge Base and the engine must somehow know when to start each one of them. To activate a particular process, it must be started by calling the `startProcess()` method in its session.

```
KieContainer kc = KieServices.Factory.get().getKieClasspathContainer();
KieSession ksession2 = kc.newKieSession("ksession-process");
session.insert(patient); //testing other session definitions
session.startProcess("com.sample.Proba"); //Start the process containing t
session.fireAllRules(); //testing other session definitions
```

4.6 Figure: Creation of a session and starting a process

The `startProcess()` method parameter represents the id of the process that needs to be started. This id needs to be specified as a property as indicated above in the part of creating the processes. If it also requires the execution of rules during the execution, it is also needed to call the `fireAllRules()` method to make sure the rules are executed as well.

Different ways to enter data into a process can be used. For example, it can be entered using the `startProcess(String processId, Map parameters)` method, which takes an additional set of parameters as name-value pairs. These parameters are copied to

the process instance as process variables. There is also the option to enter the parameters directly with the `insert(Map parameters)` method which become input data for the process.

CHAPTER 5

Use Case: Management of malnutrition in the elderly

After explaining the methodology that has been followed during this work, in this chapter the use case of the project is presented in detail.

This work is based on the NUTRIGEP project. This project aims to predict and prevent possible malnutrition in geriatric patients, managing and personalizing diets to ensure adequate and preventive nutrition. It consists of the development of a tool based on AI to prevent, monitor and generate personalized diets in order to reduce the complications caused by malnutrition (e.g., dehiscence, infections or ulcers) and which can consequently lead to a admission of the patient, which in turn means a higher health cost.

5.1 NUTRIGEP ontology

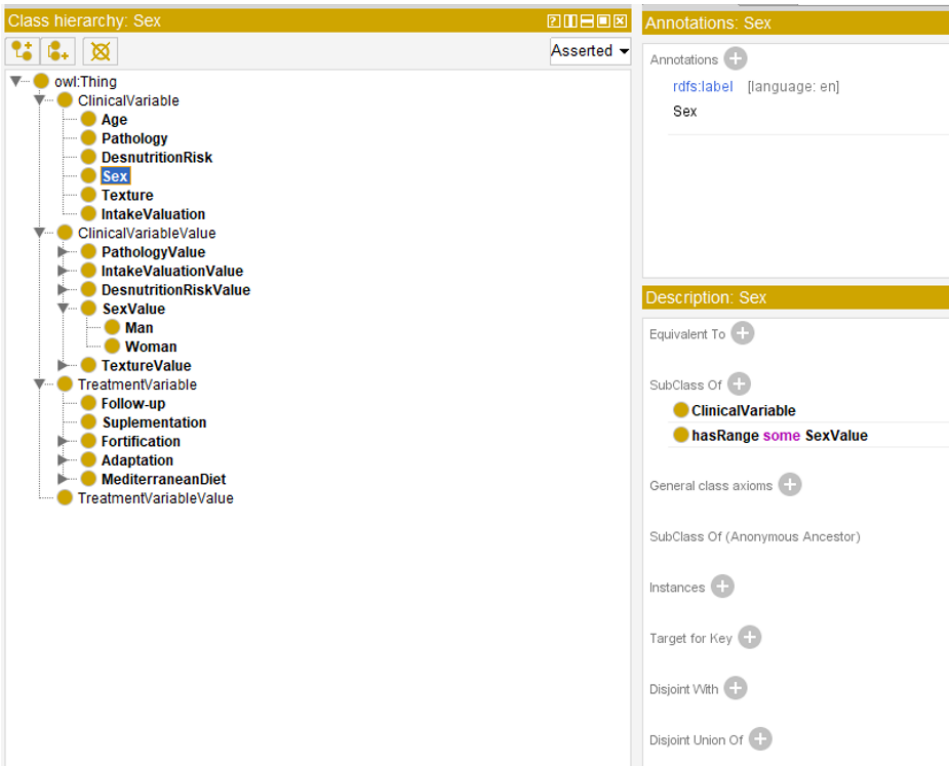
The domain of the ontology has been the malnutrition in the elderly, with the extensions of the guidelines received from nutritionists working on the project. Analyzing the guidelines, the ontology has been separated into two important groups, being (i) patient variables and (ii) treatment variables. They have been separated in this way because the treatments are personalized and depend on the patients' variables that can be introduced.

From this grouping, different classes have been determined in each group to define the variables that each group will have as can be seen in Table 5.1.

Clinical Variable	Treatment Variable
Age Sex DesnutritionRisk Pathology IntakeValuation Texture	MediterraneanDiet Fortification Adaptation Suplementation Follow-up

5.1 Table: Classes of the ontology

To specify the values that each class can have, two other groups have been created that belong, on the one hand, to the values of the classes of the clinical variables and, on the other hand, to the values of the classes of the treatment variables. With the two classes and the two groups of values created, finally the connection of them indicating the values each class can obtain have been build. Figure 5.1 shows the result of the completed ontology. In the example of the class `Sex`, the values that it will be able to obtain will only be those specified in `SexValue` that, in this case, are `Man` or `Woman`. This can be seen in the description of the Sex class that only admits having the values specified in the `SexValue` group.



5.1 Figure: NUTRIGEP ontology

5.2 Formalization of guidelines

For the knowledge base, CPGs have been translated into IF-THEN rules and formalized in several DRL files. The used CPGs came as seven excel files, each one belonging to a pathology. These guidelines were generated by nutritionists working on the project. Figure 5.2 shows an example of a treatment in the format generated by nutritionist in which they were sent to us. Each excel square belongs to a different type of patient. This ensures that, as mentioned above, the recommendations are totally personalized for each type of patient.

1. Dieta Mediterránea:

Valor Calórico Total (Kcal/día) = 2000 , Hidratos de carbono (g/día) = 249-299 de los cuales simples (g/día) < 50; Grasas (g/día) = 55-67 de las cuales saturadas (g/día) < 18, monoinsaturadas (g/día) ≤ 20-31 y poliinsaturadas (g/día) ≤ 18 g/día, colesterol (mg/día) < 300; Proteínas (g/día) = 80-100 de las cuales proteína animal (g) = 48-60 g y vegetal (g) = 32-40 (con tendencia a relación 1:1); Fibra (g/día) = 28 (20-35) de la cual soluble (g/día) = 7 e insoluble (g/día) = 21; Líquidos (l/día) = 2-2,3 y Sal (g/día) < 5.

Distribución calórica en 6 tomas (Kcal): Desayuno + almuerzo = 499 (400+100), comida + merienda = 898 (698+200) y cena + recena = 600 (500+100). Distributribución proteica (g/comida principal): 26-33; si no se alcanzaran, incluir

2. Fortificación:

En base a necesidades

2.1. Enriquecimiento calórico:

Adicionando 3 enriquecimientos al día se puede lograr un aporte extra de 150Kcal:

- Legumbres cocidas (garbanzos o judías secas) = 2 cucharas soperas (c.s)
- Frutos secos (nueces, almendra, avellana…): 1 c.s molidos/en polvo.
- Galletas tipo "María": 4 u. en polvo.
- Aceite de oliva: 1 c.s.
- Miel: 1 c.s.
- Mermelada: 1 c.s.
- Cereales de desayuno: 1 c.s en polvo.
- Salsas tipo bechamel, mayonesa o tomate frito: 1 c.s a .
- Nata líquida: 1 c.s.

2.2. Enriquecimiento proteico:

Adicionando 3 enriquecimientos al día alimentos complementarios se puede lograr un aporte extra de 16-18gr de proteína:

- Atún enlatado en aceite de oliva: 1/2 lata.
- Huevo: 1u cocida, troceada o rallada.
- Leche en polvo desnatada: 1 c.s (se calculan 10 cucharadas/l de leche).
- Yogur: adición a batidos, cereales.
- Quesitos o Queso rallado: 2 unidades y 1 c.s.
- Pescado blanco: 30gr troceado.
- Jamón cocido: 1 loncha picada.
- Pollo o pavo: 30gr. picado a.

Si no es posible llevar a cabo el enriquecimiento con alimentos naturales, se lleva a cabo artificialmente, a través de módulos proteicos (como Protifar®, Resource® Protein instant) y/o calóricos (Resource® Dextrine Maltose, Módulo ntomalt, MCT NM, Módulo de Aceite GTE, Supracal)

3.MONITOREO SEMANAL

5.2 Figure: Example of a recommendation in excel

Once the guidelines were studied, the next step was to transform them into JSON docu-

ments so that they would be interpretable by computers and easier to handle. Figure 5.3 shows the structure that guidelines have in the JSON format.

The guideline contains the attributes of the author, creation date, guideline ID and name, and a set of rule groups. Figure 5.3 also shows that the first group is the Alzheimer's pathology. Each rule group is composed of an id, a name, and a set of rules. Within the set of rules, also identified by an id and a name, there are the sets of conditions and recommendations. Figure 5.4 shows how the conditions are structured and Figure 5.5 shows the recommendations. The recommendations are separated by different orderSets that belong to the different parts of the treatment (e.g. Mediterranean diet, fortification, adaptation ...).

5.3 Figure: Structure of JSON

```
"conditionsDescription": [
    {
        "variableName": "Age",
        "variableType": "integer",
        "operator": "GREATER_THAN_OR_EQUAL_TO",
        "variableValue": 65,
        "binaryOperator": "AND",
        "childConditionsDescription": []
    },
    {
        "variableName": "Age",
        "variableType": "integer",
        "operator": "LESS_THAN_OR_EQUAL_TO",
        "variableValue": 69,
        "binaryOperator": "AND",
        "childConditionsDescription": []
    },
    {
        "variableName": "Sex",
        "variableType": "model",
        "operator": "EQUAL_TO",
        "variableValue": "Man",
        "binaryOperator": "AND",
        "childConditionsDescription": []
    },
    {…
    },
```


5.4 Figure: Conditions structure in JSON

```
"recommList": [
    {
        "orderSet": [
            {
                "type": "Diet",
                "dose": "",
                "conformance": "EQUAL_TO",
                "cycles": "",
                "duration": "2200",
                "durationUnit": "kcal/day",
                "evidenceLevel": "2A",
                "fractions": "",
                "fractionsPeriodicity": "",
                "stepNumber": "0",
                "value": "TotalCaloricValue",
                "information": ""
            },
            {…
            },
```


5.5 Figure: Recommendations structure in JSON

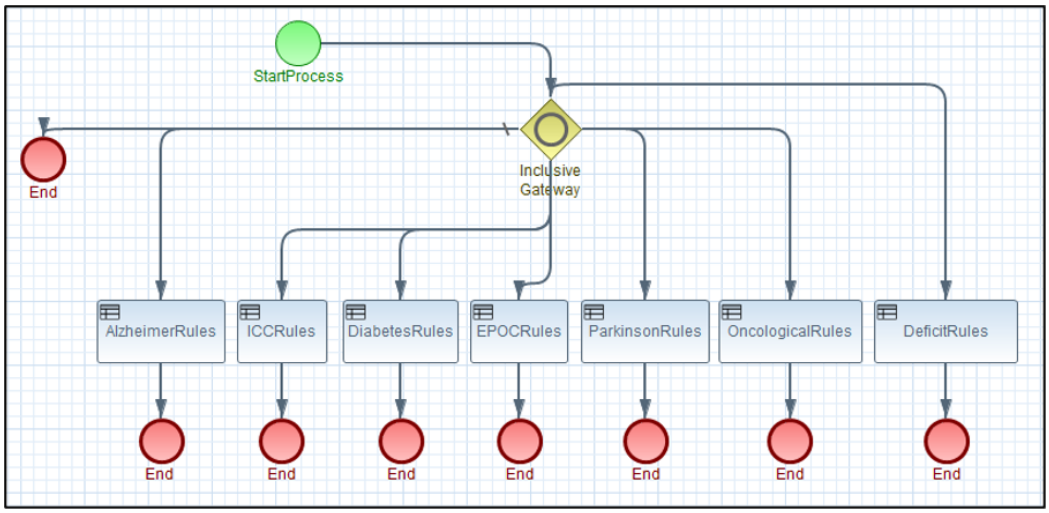
With all the guides written in JSON documents, using the Postman tool, these documents have been finally transformed into DRL files. Postman is a tool for making requests to APIs and for generating collections of requests to test them in a quick and easy way. In this way, calling the function `uploadGuidelineInDBfromGUI` the DRL files have been created. Each file contains 378 rules, being this the number of all the combinations that a patient can have. Figure 5.6 shows the final result.

```
2  package rules;
3  import java.util.ArrayList;
4  import java.util.HashMap;
5  global HashMap<String,Object> patient;
6  global ArrayList<String> recom;
7  import decisionalEventObjects.Recommendation;
8  import function debuggerHandling.DebuggingConstants.debugFiredRulesInDrools;
9  dialect "java"
10
11 rule "1 - R378"
12 ruleflow-group "alzheimer"
13 when
14     HashMap((Age >= 80.0) && (Sex == "Man") && (DesnutritionRisk == "High") && (Pathology == "Alzheimer") &&
15     (IntakeValuation == "Poor") && (Texture == "Lightly thickened") )
16 then
17     recom.add("{\"guidelineName\":\"NUTRIGEP Nutrition guideline\",\"guidelineID\":\"NUTRIGEP\",\"ruleGroupNam
18 end
19
20
21
22
23 rule "1 - R377"
24 ruleflow-group "alzheimer"
25 when
26     HashMap((Age >= 80.0) && (Sex == "Man") && (DesnutritionRisk == "High") && (Pathology == "Alzheimer") &&
27     (IntakeValuation == "Moderate") && (Texture == "Lightly thickened") )
28 then
29     recom.add("{\"guidelineName\":\"NUTRIGEP Nutrition guideline\",\"guidelineID\":\"NUTRIGEP\",\"ruleGroupNam
30 end
31
32
33
34
35 rule "1 - R376"
36 ruleflow-group "alzheimer"
37 when
38     HashMap((Age >= 80.0) && (Sex == "Man") && (DesnutritionRisk == "High") && (Pathology == "Alzheimer") &&
39     (IntakeValuation == "Appropiate") && (Texture == "Lightly thickened") )
40 then
41     recom.add("{\"guidelineName\":\"NUTRIGEP Nutrition guideline\",\"guidelineID\":\"NUTRIGEP\",\"ruleGroupNam
```

5.6 Figure: Final DRL file

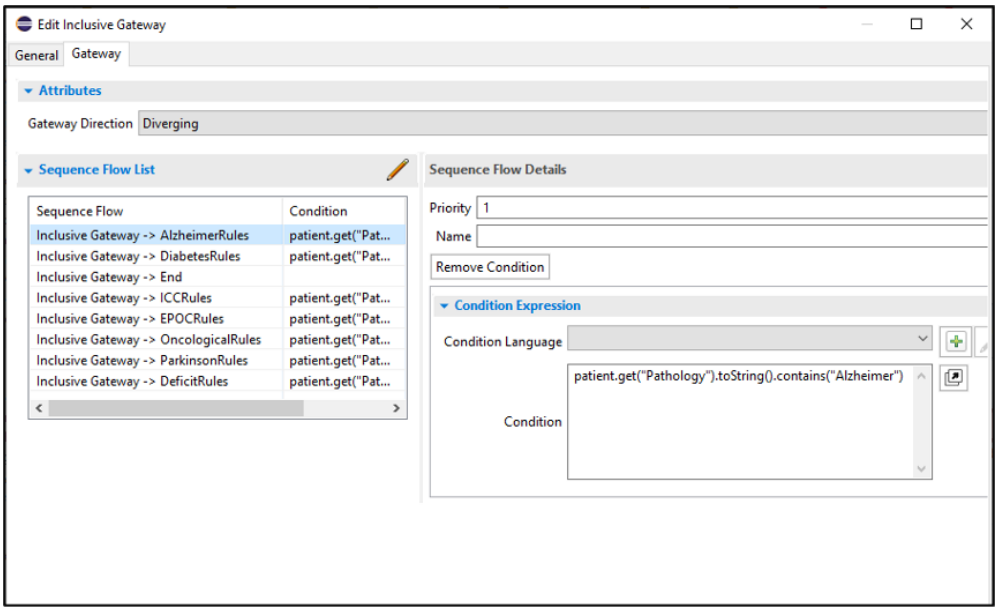
5.3 Creation of the process

With all the guidelines formalized in DRL files, the last step consisted of integrating the decision flows in the code for the management of the seven DRLs. Figure 5.7 shows the process that has been created.



5.7 Figure: DRL management process

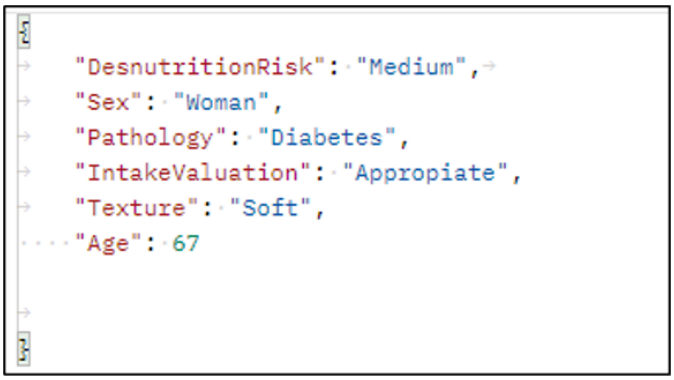
The process starts with a start node called `StartProcess`. This is where the patient characteristics are received. Following the path below a gateway is encountered which in this case is inclusive. The reason why it is inclusive is as a patient can be multi-pathological, so the patient should receive more than one recommendation. Taking into account that DRLs have been separated by pathology, the gateway has been separated into seven different paths, each one in the direction of a group of rules belonging to the same pathology. These groups of rules are called Rule Tasks and are linked to the DRLs thanks to the Rule Flow Group attribute, so that only the rules related to the pathologies suffered by the patient are evaluate. Figure 5.8 shows how in the characteristic panel of the door, the patient must fulfill a condition to continue along the paths. In the example it is observed how the patient will arrive at the Alzheimer's Rule Task if he/she respects to the condition corresponding to the patient's characteristic called "Pathology".



5.8 Figure: Gateway characteristics panel

5.4 Patient type

To enter a patient into the project, the Postman tool has also been used, calling the function `getRecommendationsForPatient` and entering a patient in JSON format as shown in Figure 5.9.



5.9 Figure: Patient type

For this same example, Figure 5.10 illustrates the recommendations received belong to the Diabetes rule group and exactly rule number 4 has been executed.

```
{
    "PatientRecom": [
        {
            "guidelineName": "NUTRIGEP Nutrition guideline",
            "ruleGroupName": "Diabetes",
            "guidelineID": "NUTRIGEP",
            "ruleGroupID": "NUTRIGEP_RG_02",
            "recommendations": [
                {
                    "usability": 0.78,
                    "orderSet": [...
                    ],
                    "sourceRuleID": "NUTRIGEP_RG_02_RL_04",
                    "strengthOfRecomm": 0.61
                }
            ]
        }
    ]
}
```


5.10 Figure: Recommendation with one pathology

In case the patient is multi-pathological, the attribute Pathology will be formed by an array, for example [Alzheimer, Diabetes]. So, if one more pathology is added to the patient, for example Alzheimer's disease, the result would be as shown in Figure 5.11. It can be seen that, apart from the previous recommendation, a new one is added by the Alzheimer's rule group. That way the patient can receive the recommendations of both diseases.

```
{
    "PatientRecom": [
        {
            "guidelineName": "NUTRIGEP Nutrition guideline",
            "ruleGroupName": "Alzheimer",
            "guidelineID": "NUTRIGEP",
            "ruleGroupID": "NUTRIGEP_RG_01",
            "recommendations": [
                {
                    "usability": 0.19,
                    "orderSet": [...
                    ],
                    "sourceRuleID": "NUTRIGEP_RG_01_RL_04",
                    "strengthOfRecomm": 0.93
                }
            ]
        },
        {
            "guidelineName": "NUTRIGEP Nutrition guideline",
            "ruleGroupName": "Diabetes",
            "guidelineID": "NUTRIGEP",
            "ruleGroupID": "NUTRIGEP_RG_02",
            "recommendations": [...
            ]
        }
    ]
}
```


5.11 Figure: Recommendations with more than one pathology

CHAPTER 6

Conclusions

After realizing the work, several conclusions have been drawn, at academic and personal level. In addition, several possible improvements or changes for the future have been analyzed.

6.1 Conclusions

6.1.1 Objectives achieved

During the development of the work, the initially proposed objectives have been achieved. On the one hand, an exhaustive analysis of the current landscape of clinical practice guidelines in both paper and digital format has been carried out, including the analysis of the operation of rules engines and the management of the guidelines within them. Real guidelines have been obtained from professionals in the field of malnutrition and have been formalized in digital format. In addition, a domain ontology centered in the nutrition domain has been created to restrict the semantic problems that may occur. Finally, a deep understanding of Drools and decision flows have been developed to create a solution to manage the guidelines and give personalized recommendations. After an on-site analysis of the state-of-the-art, it has been shown that flows allow the management of different DRL, which in real clinical practice it translates in the management of clinical guidelines, allowing to evaluate clinical data against a set of different CIGs simultaneously.

6.1.2 Personal reflection

This work has meant a change in my academic career since I have joined a company and it has enriched me a lot by the fact of working on projects that are in operation now.

On a personal level, the development of this work has been very satisfactory. On the one hand, I am satisfied for having understood the functionality of the software that I was given at the beginning on an individual way and for the constancy of the work done to achieve the objectives, proposing different solutions along the way. On the other hand, the organization of the project has been very orderly due to the experience of my managers in this type of work, focusing from the beginning on a goal, without letting me get sidetracked.

Academically, doing the project in a technology park made me feel that I was part of the company and that my work had the same importance and value as others. It also made me realize that there are infinite areas that a computer science can contribute in different ways. It has also made me realize that what I have learned in my career can be used to work in different fields. In my case, I have noticed it when handling the java language, since my project is based on a java structure and artificial intelligence. But without any doubt, the most enriching thing has been to work with people with different backgrounds, most of them had studied biomedical engineering. Besides, it has been very positive to see that we could help each other a lot to carry out the project even though the situation has not been the easiest due to the fact that I had to work telematically.

Finally, it is worth mentioning that this work has made me evolve both academically and personally. Working on an existing project has increased my motivation to do it and I took it as a challenge without having met anyone at the beginning. But finally, I am very grateful for the adaptation and satisfied with the results I achieved.

6.2 Future Work

This chapter presents several open doors to be explored in the future to improve and expand the work done:

- *Create RuleFlow processes using the Process API*: As mentioned before, there are different ways to create processes. In this work, processes have been created using the most visual approach using the graphical RuleFlow editor, but this can become

a problem to share it because it might not be compatible with other programs. The Process API allows us to configure flows outside Eclipse. This way, anyone could generate specific flows from a user-friendly web-based application, thus no specific software is needed to generate the same flows. At this point it is necessary to have Eclipse, to know how flows work and to have the help of an expert in the area. In the future, it will be integrated in a process authoring tool, so that it will not be necessary to have to install specific software and an expert is not needed to configure a flow.

- *Validation*: a validation of the platform should be performed in order to ensure (i) the correct execution of the configured processes and (ii) the clinical value of the obtained recommendations. For this purpose, ideally, a study should be performed in which a specialist (a nutritionist in our case) would use the developed system in order to obtain recommendations, and then the specialist would indicate whether to accept or not the obtained recommendation. In the future, it could even be possible to test different patients in compliance with the recommendations and see if they really improve the patient's nutritional status. This way, by looking if the patient's condition improves because of the proposed nutritional recommendations, it could be possible to assess the clinical value of the formalized nutritional recommendations, and that the developed platform is useful in a real-world scenario.

Bibliography

[1] P. Alonso and X. Bonfill. Clinical practice guidelines (I): Elaboration, implementation and evaluation. *Radiologia*, 49:19–22, 2007.

[2] R. Harbour and J. Miller. A new system for grading recommendations in evidence-based guidelines. *British Medical Journal*, 2:115–120, 1924.

[3] M. Peleg. Computer-interpretable clinical guidelines: A methodological review. *Journal of Biomedical Informatics*, 46:744–763, 2013.

[4] M. A. Navarro Puerto et al. ¿Las guías que nos guían son fiables? Evaluación de las guías de práctica clínica españolas. *Revista Clínica Española*, 205:533–540, 2005.

[5] P. Gillois, G. Chatellier, M. C. Jaulent, et al. From paper-based to electronic guidelines: Application to french guidelines. *Studies in Health Technology and Informatics*, 84:196–200, 2001.

[6] A. M. Shahsavarani, E. Azad, M. Abadi, and M. H. Kalkhoran. Cinical Decision Support Systems (CDSSs): State of the art Review of Literature. *International Journal of Medical Reviews*, 2:299–308, 2015.

[7] P. de Clercq, K. Kaiser, and A. Hasman. Computer-interpretable Guidelines Formalisms. *Stud Health Technol Inform*, 139:22–43, 2008.

[8] A. Latoszek-Berendsen, H. Tange, H. J. Van de Herik, and A. Hasman. From clinical practice guidelines to computer-interpretable guidelines: A literature overview. *Methods of Information in Medicine*, 49:550–570, 2010.

[9] M. Peleg et al. Comparing Guideline Models: A Case-study Approach. *Journal of the American Medical Informatics Association*, 10:52–68, 2003.

[10] R. Minor. Computer-interpretable guidelines using GLIF with windows workflow fundation. Master's thesis, The school of Graduate Studies, Laurentian University, 2014.

[11] A. Boxwala et al. GLIF3: A representation format for sharable computer-interpretable clinical practice guidelines. *Journal of Biomedical Informatics*, 37:147–161, 2004.

[12] A. Sarri and Y. Carri. Un acercamiento a las ontologías médicas y su importancia. Master's thesis, Centro de Información Médica, Cuba, 2016.

[13] M. Teresa Romá-Ferri and M. Palomar. Análisis de terminologías de salud para su utilización como ontologías computacionales en los sistemas de información clínicos. *Gaceta Sanitaria*, 22:421–433, 2008.

[14] G. Fragoso, S. de Coronado, M. Haber, F. Hartel, and L. Wright. Overview and utilization of the NCI Thesaurus. *Comparative and Functional Genomics*, 5:648–654, 2004.

[15] Y. Castillo Quiel, A. Saavedra, and V. Villarreal. Estándares de codificación e interoperabilidad en Salud: evaluación del proyecto AmIHEALTH. *Revista Cubana de Información en Ciencias de la Salud*, 30:1–13, 2019.

[16] R. Miller, L. Waitman, and S. Rosenbloom. *Decision Support During Impatient Care Provider Order Entry: Vanderbilt Experience*. 2016.

[17] J. Lee, J. Kim, I. Cho, and Y. Kim. Integration of workflow and rule engines for clinical decisions support services. *Studies in Health Technology and Informatics*, 160:811–815, 2010.

[18] Vimla L. Patel, Vanessa Allen, J. F. Arocha, and Edward H. Shortiffle. Representing Clinical Guidelines in GLIF: Individual and Collective Expertise. *Journal of the American Medical Informatics Association*, 5:467–483, 1998.

[19] Y. Pacheco Cárdenas, S. Estévez Abrahantes, and M.E. Martínez del Busto. Integración de un Sistema de Gestión de Reglas de negocio al flujo de trabajo control de historias clínicas para trasplante renal. *Rev. Cuba inf méd*, 7:105–112, 2015.

[20] N. Muro et al. Architecture for a Multimodal and Domain-Independent Clinical Decision Support System Software Development Kit. *Proceedings of the Annual*

International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS, pages 1399–1404, 2019.

[21] Marcelo Martín Marciszack, Manuel Pérez Cota, Rubén Leandro Antonelli, Roxana Silvia Giandini, and Marina E Cardenas. Construcción de una ontología para gramáticas formales y máquinas abstractas utilizando protégé para la elicitación de requerimientos. In *XI Workshop de Investigadores en Ciencias de la Computación*, 2009.

[22] D.J.V. García. Estudio de viabilidad de Dools-Guvnor para su integracion en el simulador empresarial SIMBA. Master's thesis, Universidad Carlos III de Madrid, 2012.